UNISYS

MAPPER® System Run Design

Operations Reference Manual

Volume 2: Run Statements

May 1990

Printed in U S America
7831 9274–000

Priced Item

UNISYS

MAPPER® System Run Design

Operations Reference Manual

Volume 2: Run Statements

Copyright © 1990 Unisys Corporation
All Rights Reserved
Unisys and MAPPER are registered trademarks of Unisys Corporation

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 financial 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

Contents

Volume 2: Run Statements

Section 7. Run Statements

Figures

Tables

Section 7
Run Statements

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

This section presents the MAPPER run statements by run function call in alphabetical order. Complete reference information is included about most statements; those that are documented only in the online help system are flagged with an asterisk in Table 7-1.

If you need tutorial information, refer to the *Run Design Training Guide*.

This section contains:

- A list of run statements by name
- A list of run statements by topic
- Reference information for run statements, presented alphabetically

Run Statements by Name

In Table 7-1, the descriptive run statement names and their corresponding run function calls are listed for cross-reference purposes. Some MAPPER statements are documented in the online help system only. These are flagged with an asterisk (*) in Table 7-1.

Note: *Since this list encompasses several different kinds of MAPPER systems, some run statements may not be available on your system.*

Table 7-1. Run Statements by Name

Name	Call
A Series Run	ASR
Acknowledge Message	OK
Add Line	LN+ *
Add Report	ADR *
Append Line	LNA *
Append Report	ADD *
Arithmetic	ART
Auxiliary	AUX
Background Run	BR
Binary Find	BFN
Break	BRK
Break Graphics	BRG *
Build Label Table	BLT *
Cabinet Switch	CAB *
Cache Report	CAH *
Calculate	CAL
Calculate Update	CAU *
Call Subroutine	CALL
Change Variable	CHG
Clear Abort Routine	CAR *
Clear Error Routine	CER *
Clear Label Table	CLT *
Clear Link	CLK *
Clear Subroutine	CSR *
Clear Variables	CLV *
Command Handler	CHD

continued

* Documented in online help system only (enter **help,@*call***).

Table 7-1. Run Statements by Name (cont.)

Name	Call
Commit Updates	CMU *
Compare Report	CMP
Copy	CPY *
Count	CNT
Create Result Copy	RSL *
Create File	FIL
Data Definition Information	DDI †
Date	DAT
Date Calculator	DC
DDP Copy	DCPY *
DDP Create	DCRE *
DDP Purge	DPUR *
Decode Report	DCR
Decommit Updates	DCU *
Decrement Variable	DEC *
Defer Updates	DFU *
Define	DEF
Define Variable Size	DVS
Delete	DEL *
Delete Line	LN- *
Delete Report	DLR *
Device	DEV *
Directory	DIR
Diskette	DIS *
Display Form	DSF
Display Graphics	DSG *
Display Message	DSM
Display Report	DSP
Display Report/Exit	DSX *
Downline Load	DLL *
Drawer	DRW
Duplicate Line	LNX *
Duplicate Report	DUP *
Element	ELT
Element Delete	EL-
Encode Report	ECR
Exchange Variables	XCH *
Execute	XQT

* Documented in online help system only (enter **help,@*call***).
† Documented in *MRI Run Statements Reference.*

Table 7-1. Run Statements by Name (cont.)

Name	Call
Exit MAPPER System	XUN
Exit Subroutine	ESR *
Extract	EXT *
Find	FND
Find and Read Line	FDR
Format	FMT
Function Key	FKY
Function Key Input	KEY
Generate Organization Chart	GOC *
Go To	GTO
Graphics Scaler	GS *
Hash	HSH
Host Read	HRD *
Host Run	HRN *
Host Sign-off	HOF *
Host Sign-on	HST *
Host Write	HWR *
If Conditional	IF
Increment Variable	INC *
Index	IND
Index User	IDU
Input Variable	ITV
Insert Line	LNI *
Insert Variable	INS *
Justify Variable	JUV
Language	LNG *
Last Line Number	LLN *
Line Zero	LZR
Link to Another Run	LNK
List Merge	LMG *
Load Field Name	LFN
Load Format Characters	LFC
Load System Message	LSM
Load Variable	LDV
Load Variable Array	LDA

* Documented in online help system only (enter **help,@*call***).

Table 7-1. Run Statements by Name (cont.)

Name	Call
Local Run Call	**cmd*
Locate	LOC
Locate and Change	LCH
Locate and Change Variable	LCV
Log for Analysis	LOG * †
Logoff Relational System	LGF †
Logon Relational System	LGN †
Match	MCH
Match Update	MAU *
Message to Console	MSG
Move Line	LNM *
Network Off	NOF
Network Read	NRD
Network Remote	NRM
Network Return	NRT
Network Run	NRN
Network Sign-On	NET
Network Write	NWR
Operating System Interface	OS2
Out Variable	OUV
Output	OUT
Output Mask	OUM
Peek Variables	PEK *
Poke Variables	POK *
Pop Variables	POP *
Print	PRT
Push Variables	PSH *
Put Line	LNP *
Queue Control	QCTL *
Read Continuous	RDC
Read Line	RDL
Read Line Next	RLN
Read Password	RPW
Reformat Report	RFM

* Documented in online help system only (enter **help,@*call***).
† Documented in *MRI Run Statements Reference.*

Table 7-1. Run Statements by Name (cont.)

Name	Call
Refresh Screen	PNT *
Register Abort Routine	RAR
Register Error Routine	RER †
Relational Aggregate Fetch	FCH †
Relational Aggregate Modify	RAM †
Release	REL *
Release Message	QREL *
Remote Run	RRN *
Remote Symbiont Interface	RSI
Remove Variables	RMV *
Rename	RNM *
Replace Report	REP *
Retrieve File	RET
Retrieve from History	REH
Return Call Routine	RETURN *
Return Remote	RTN *
Run Debug	RDB
Run Start	RUN
Run Status	RS
Run Subroutine	RSR
Schedule	SCH
Screen Control	SC
Search	SRH
Search Update	SRU *
Send Message, No Response	QSND *
Send Message, Expect Response	QSNR *
Send Report	SEN
Send Report to User	SNU
Send Response Message	QRSP *
Set Format Characters	SFC
Sign off	XIT *
Sort	SOR
Sort and Replace Report	SRR *
Start	STR
Station Information	STN †
Submit SQL Statement	SQL †
Subtotal	SUB
Tape Cassette	TCS *

* Documented in online help system only (enter **help,@*call***).
† Documented in *MRI Run Statements Reference*.

Table 7-1. Run Statements by Name (cont.)

Name	Call
Totalize	TOT †
Trace	TRC †
UNIX Interface	UNX
Unlock	ULK *
Update	UPD *
Update Lock	LOK *
Use Variable Name	USE
Wait	WAT
Word Change	WDC *
Word Locate	WDL *
Word Process	WPR *
Write Line	WRL
Yank Line	LNY *

* Documented in online help system only (enter **help,@*call***).
† Documented in *MRI Run Statements Reference.*

Run Statements by Topic

This section categorizes MAPPER run statements and describes them briefly.

Note: Since this list encompasses several different kinds of MAPPER systems, some run statements may not be available on your system.

Locating, Changing, Comparing, Sorting, and Reformatting Data

The following statements locate, change, compare, sort, or reformat data. See also "Manipulating and Updating Report Lines" in this subsection.

- Binary Find (BFN) - Finds data items in a sorted report by sampling the report from the midpoint.
- Compare Report (CMP) - Compares the contents of a report or result with another report.
- Find (FND) - Finds data in a report or result.
- Find and Read Line (FDR) - Finds data in a report or result.
- Generate Organization Chart (GOC) - Generates organization charts using the GOC command language.
- Locate (LOC) - Locates a character string within a report or result.
- Locate and Change (LCH) - Searches for a specific target string and replaces it with another specified string in a report or result.
- Match (MCH) - Matches fields in two reports, and optionally moves data from an issuing report to a receiving report, creating a result.
- Match Update (MAU) - Performs the same operation as MCH, but you can blend the changed lines back into the original report (cannot be a result). The receiving report is locked so that it cannot be updated while the match is being performed.
- Read Continuous (RDC) - Reads report lines and segments into the output area.
- Reformat Report (RFM) - Moves data between reports or across drawers and creates a result.

- Search (SRH) - Searches vertically through reports or results and puts located items in a result.
- Search Update (SRU) - Does the same thing as SRH, but you can also blend the changed lines back into the original report (cannot be used on a result).
- Sort (SOR) - Sorts report data.
- Sort and Replace Report (SRR) - Sorts report data and replaces the sorted data into the original report (cannot be used on results).

Manipulating Reports and Results

The following statements manipulate reports and results:

- Add Report (ADR) - Adds a new report to a drawer.
- Append Report (ADD) - Appends a report or result to the end of another report or result.
- Break (BRK) - Turns the output area of a run into the current -0 result and creates a new output area (containing only a date line).
- Break Graphics (BRG) - Packs data, such as graphics primitive code, in the output area and places it into a result.
- Cache Report (CAH) - Loads (caches) all of the specified report into memory for fast processing.
- Create Result Copy (RSL) - Creates a copy of a report as a result.
- Decode Report (DCR) - Transforms an encoded report into a readable report by specifying the key.
- Delete Report (DLR) - Deletes a MAPPER report.
- Duplicate Report (DUP) - Copies an existing report or result, creating a new report.
- Encode Report (ECR) - Encodes a report, changing the data from readable text into code that can be read only if a key is specified.
- Generate Organization Chart (GOC) - Generates organization charts using the GOC command language.

- Rename (RNM) - Creates a new reference for a report or result, -1 through -8, in any order.
- Replace Report (REP) - Replaces the contents of a report with the contents of another report or result.

Manipulating Screen Displays

The following statements manipulate screen displays:

- Change Variable (CHG INPUT$) - Loads variables with input from a screen display.
- Command Handler (CHD) - Registers a routine to be executed whenever the user of the run enters information in the control line after a DSP, OUT, or SC statement.
- Display Form (DSF) - Displays a report containing screen control commands (form) from within a run.
- Display Graphics (DSG) - Translates the graphics primitive code in a report and displays it on your terminal.
- Display Message (DSM) - Allows you to display your own one-line message at the top of the screen.
- Display Report (DSP) - Displays a report or result on your terminal.
- Display Report and Exit (DSX) - Displays a report or result on your terminal and exits the run.
- Format (FMT) - Creates a display format by selecting which fields of a report or result to display on a following DSP, OUM, or OUT statement.
- Function Key (FKY) - Stores function key information for a function key bar and loads it into the reserved word FKEY$.
- Graphics Scaler (GS) - Increases or decreases the size of your charts and creates custom graphics using the expanded syntax.
- Input Variable (ITV) - Accepts input into variables from a screen displayed with an Out Variable (OUV) statement.
- Language (LNG) - Lets you switch to another language for system messages.

- Load Format Characters (LFC) - Captures the display format of a report.
- Out Variable (OUV) - Displays the contents of variables, reserved words, constants, or literal data on the screen.
- Output (OUT) - Displays portions of a report or result or the output area of a run.
- Output Mask (OUM) - Displays a blank function mask from report headings.
- Refresh Screen (PNT) - Refreshes the screen display.
- Release Display (REL) - Terminates the run and displays the active screen.
- Screen Control (SC) - Allows you to create menus and other input screens or to edit text already on the screen. You can also use it to overlay existing OUT or DSP screens.
- Set Format Characters (SFC) - Defines a format for the display of a report or result with a DSP, DSX, OUM, or OUT statement.
- Sign off MAPPER Software (XIT) - Terminates a MAPPER session and signs off the user (sign-on screen).
- Wait (WAT) - Suspends a screen display.

Obtaining Information about the Database, Display, and Runs

The following statements provide information about the database, the display terminal, and runs currently being executed:

- Directory (DIR) - Loads variables with information about a data name from the system directory.
- Drawer (DRW) - Loads a variable with the number of characters per line in a drawer.
- Index (IND) - Indexes a specific drawer in a cabinet and creates a result.
- Index User (IDU) - Indexes a specific drawer in a cabinet and creates a result by user, date, or report range.

- Last Line Number (LLN) - Loads a variable with the last line number of a specified report.
- Line Zero (LZR) - Initializes or loads variables with data from line 0 of a report or result.
- Load Field Name (LFN) - Loads variables with the names of report fields that correspond to the column-character positions supplied.
- Load Format Characters (LFC) - Captures the display format of a report.
- Log for Analysis (LOG) - Logs statements executed in a run.
- Retrieve from History (REH) - Retrieves the version of a MAPPER report prior to the last system purge or merge process.
- Run Status (RS) - Creates a result showing the status of runs being executed with your user-id.

Calculating

The following statements perform calculations on literal data, report data, or dates and times:

- Arithmetic (ART) - Performs arithmetic on variables or constants.
- Calculate (CAL) - Performs arithmetic calculations on a report or result.
- Calculate Update (CAU) - Performs the same operation as CAL, but you can also blend the changed lines back into the original report.
- Count (CNT) - Computes subtotals, percentages, standard deviations, averages, entry counts, and other calculations on reports or results, creating a result.
- Date (DAT) - Computes dates in a report or result.
- Date Calculator (DC) - Computes and calculates dates and times in variables.
- Subtotal (SUB) - Subtotals report or result data.
- Totalize (TOT) - Performs arithmetic and move operations on report or result fields.

Printing Results or Reports

The following statements print reports and results on auxiliary or system printers:

- Auxiliary (AUX) - Sends a report or result to auxiliary printers.
- Print (PRT) - Prints reports or results on the system printer.

Manipulating Variables

The following statements manipulate variables in several ways:

- Change Variable (CHG) - Changes or initializes the contents of a variable.
- Clear Variables (CLV) - Clears the definition and contents of a set of variables.
- Decrement Variable (DEC) - Decrements the numeric value of a variable.
- Define (DEF) - Tests the contents and type of a variable or array.
- Define Variable Size (DVS) - Creates variables equal to the size of report fields.
- Exchange Variables (XCH) - Exchanges a set of variables in your run with a set stored in a level of the variable stack.
- Increment Variable (INC) - Increases the numeric value of a variable.
- Input Variable (ITV) - Accepts all input from OUV or interim OUT output and initializes variables.
- Insert Variable (INS) - Inserts variable or constant data into a variable.
- Justify Variable (JUV) - Reformats the contents of numeric variables.
- Load Variable (LDV) - Loads or initializes variables with specified contents.
- Load Variable Array (LDA) - Defines a variable array and puts data into the array.
- Locate and Change Variable (LCV) - Locates a string in a variable and optionally changes the string.

- Out Variable (OUV) - Displays the contents of a variable, constant, reserved word, or literal data.
- Peek Variables (PEK) - Retrieves variables saved with a previous Push (PSH) statement.
- Poke Variables (POK) - Replaces the contents and definition of variables saved with a previous Push (PSH) statement.
- Pop Variables (POP) - Retrieves variables saved with a previous Push (PSH) statement and removes their level from the stack.
- Push Variables (PSH) - Saves both the contents and definition of variables so that you can restore them later in the run with a POP or a PEK statement.
- Remove Variables (RMV) - Removes a level from the variable stack.
- Use Variable Name (USE) - Allows you to assign a variable name to a specific variable number.

Controlling Runs

The following statements encompass several areas of run design, including logic, subroutines, and utilities:

- Background Run (BR) - Begins a run in the background so that you can still perform operations from the terminal.
- Break (BRK) - Puts the output area of a run into a result (-0).
- Build Label Table (BLT) - Builds label table definition lines for a run control report.
- Cabinet Switch (CAB) - Switches the current cabinet to a different cabinet.
- Call Subroutine (CALL) - Saves the contents and definitions of all variables currently defined in the run and transfers control to a subroutine; you can also pass the value of up to 40 variables to the called subroutine.
- Clear Abort Routine (CAR) - Cancels an abort routine set by a Register Abort Routine (RAR) statement.

- Clear Error Routine (CER) - Cancels an error routine set by a Register Error Routine (RER) statement.
- Clear Link (CLK) - Clears the link to the original run from the run that was started by a LNK statement.
- Clear Label Table (CLT) - Deletes label table definition lines from a run control report.
- Clear Subroutine (CSR) - Clears the return path to the calling run control report in an external subroutine called by the Run Subroutine (RSR) statement.
- Command Handler (CHD) - Registers a routine to be executed whenever the user of the run enters information in the control line after a DSP, OUT, or SC statement.
- Execute (XQT) - Executes a run statement.
- Exit Subroutine (ESR) - Exits a subroutine and returns you to the line following the RSR statement in the run control report.
- Function Key Input (KEY) - Provides you with a number indicating the function key the run user pressed after a noninterim display.
- Go To (GTO) - Allows you to branch within a run (GTO *label* or LIN *n*), execute run statements in another run control report (GTO RPX), or terminate a run and display the output area (GTO END).
- If Conditional (IF) - Compares values and performs an action depending on whether the condition is true or false.
- Link to Another Run (LNK) - Executes another run in the MAPPER system.
- Load System Message (LSM) - Loads a variable with the contents of a system message.
- Read Password (RPW) - Allows a run to access reports containing one of two read passwords.
- Register Abort Routine (RAR) - Registers or names a routine specified by a label to be executed if the user aborts a run.
- Register Error Routine (RER) - Registers or names a routine specified by a label to be executed if the run encounters an error.

- Return Call Routine (RETURN) - Exits the called subroutine and returns to the calling run.
- Run Debug (RDB) - Allows you to step through a run, one statement at a time, to debug it.
- Run Start (RUN) - Terminates the current run and executes another run at the same station.
- Run Subroutine (RSR) - Transfers control to a specified label to execute a set of run statements.
- Schedule (SCH) - Allows you to schedule any queuing function that follows the SCH statement on the same logic line.
- Wait (WAT) - Temporarily suspends the execution of a run.

Manipulating and Updating Report Lines

The following statements allow you to manipulate or update reports and results:

- Add Line (LN+) - Adds lines to a report or result.
- Append Line (LNA) - Adds lines to an existing temporary buffer.
- Commit Updates (CMU) - Commits updates to reports that were deferred.
- Decommit Updates (DCU) - Decommits updates to reports which were deferred.
- Defer Updates (DFU) - Defers all updates to specified reports until updates are committed or decommitted.
- Delete (DEL) - Deletes lines in an update result (SRU, MAU, CAU) from the original report.
- Delete Line (LN-) - Deletes lines from a report or result.
- Duplicate Line (LNX) - Duplicates lines within a report or result.

- Extract (EXT) - Extracts lines in an update result from a report, deletes them from the original report, and moves the extracted lines into the current result.
- Insert Line (LNI) - Inserts a line or lines into a report or result.
- Move Line (LNM) - Moves lines within a report or result.
- Put Line (LNP) - Copies lines from a temporary buffer into a report or result.
- Read Continuous (RDC) - Reads report lines and segments into the output area.
- Read Line (RDL) - Reads a report line or line segment.
- Read Line Next (RLN) - Continues reading from the report or current result after an RDL or FDR.
- Unlock (ULK) - Releases an update lock on a report.
- Update (UPD) - Blends updated lines in the result of an update result (SRU, MAU, CAU) back into the report.
- Update Lock (LOK) - Assumes update control of a report, preventing other users from updating it until it is released.
- Write Line (WRL) - Updates up to 23 lines of a report or result.
- Yank Line (LNY) - Copies lines from a report or result into a temporary buffer area.

Sending and Receiving Messages, and Handling Devices

The following statements allow you to send and receive messages and handle devices:

- Acknowledge Message (OK) - Acknowledges an incoming message.
- Device (DEV) - Lists auxiliary devices if any are connected to a terminal.
- Diskette (DIS) - Writes data to and reads data from a 5-1/4-inch diskette drive connected to a Unisys UTS 30 display terminal.

- Downline Load (DLL) - Loads a precompiled program stored in a MAPPER report into a Unisys UTS 400 master or UTS 40 display terminal.
- Message to Console (MSG) - Sends a message to the system console.
- Send Report (SEN) - Sends a report or result as a message to a station as a result.
- Send Report to User (SNU) - Sends an entire report or result to another user.
- Station Information (STN) - Provides you with information about a specific station number.
- Tape Cassette (TCS) - Transfers data to and from a tape cassette/diskette unit connected as an auxiliary device of a display terminal.

Interfacing the Operating System, Other Applications, and Other MAPPER Systems

The following statements allow you to work with the operating system, applications outside of MAPPER software, and with other MAPPER systems.

Note: See the online help system (HELP,DOC) for a list of related product information. Some of the run statements listed in this category are described in other manuals.

- A Series Run (ASR) - Enables you to execute A series programs.
- Copy (CPY) - Copies an OS 1100 program file or element, or a data file, from one site to another through the remote run link.
- Create File (FIL) - Creates a native data file from a MAPPER report or result.
- Data Definition Information (DDI) - Retrieves a table description from a relational database.
- DDP Copy (DCPY) - Copies an OS 1100 program file or element, or data file, from one host to another using DDP 1100.

- DDP Create (DCRE) - Creates a file on a DDP 1100 host.
- DDP Purge (DPUR) - Deletes a file on a DDP 1100 host.
- Element (ELT) - Copies a MAPPER report or result to a standard OS 1100 program file or symbolic element, or to a data file.
- Element Delete (EL-) - Deletes a standard OS 1100 program file or symbolic element, or a data file.
- Exit MAPPER System (XUN) - Terminates a MAPPER session, exits MAPPER software, and returns you to your previous environment.
- Host Read (HRD) - Lets you transfer data from the OS 1100 MAPPER System.
- Host Run (HRN) - Lets you transfer run statements to the OS 1100 MAPPER System to be executed.
- Host Sign-off (HOF) - Signs you off the OS 1100 MAPPER System.
- Host Sign-on (HST) - Signs you on to the OS 1100 MAPPER System.
- Host Write (HWR) - Lets you write a report or result to the OS 1100 MAPPER System.
- Logoff Relational System (LGF) -Terminates communication with a relational database manager.
- Logon Relational System (LGN) - Establishes communication between a MAPPER system and a relational database management system.
- Network Off (NOF) - Signs the local MAPPER system caller off the remote MAPPER system.
- Network Read (NRD) - Enables the user to read and pass data from a remote MAPPER system to the local MAPPER system.
- Network Remote (NRM) - Executes a run that resides on a remote MAPPER system.
- Network Run (NRN) - Allows a local MAPPER system user to pass run statements to a remote MAPPER system to be executed.
- Network Return (NRT) - Returns control from a run executed with NRM to the run executing on the local system.

- Network Sign On (NET) - Signs on to the remote MAPPER system from the calling (local) MAPPER system.
- Network Write (NWR) - Allows a user to send a report or result from a local MAPPER system to a remote MAPPER system.
- Operating System Interface (OS2) - Interfaces with the operating system and allows the execution of native PC operating system commands.
- Queue Control (QCTL) - Obtains various information about the Data Transfer Module (DTM) interface.
- Relational Aggregate Fetch (FCH) - Retrieves data from one or more relational tables and places the data in a MAPPER result.
- Relational Aggregate Modify (RAM) - Creates or deletes relational tables, updates a table, or issues a COMMIT command.
- Release Message (QREL) - Indicates that Data Transfer Module (DTM) processing is complete.
- Remote Run (RRN) - Starts a run in background at another MAPPER site.
- Remote Symbiont Interface (RSI) - Initiates an interactive demand program through a MAPPER run.
- Retrieve File (RET) - Retrieves a native data file as a MAPPER result.
- Return Remote (RTN) - Returns a report or result from a remote MAPPER site to the local site as a current result.
- Send Message, No Response (QSND) - Through Data Transfer Module (DTM), sends a message to a named queue, expecting no response.
- Send Message, Response Expected (QSNR) - Through Data Transfer Module (DTM), sends a message on a named queue, expecting a response.

- Send Response Message (QRSP) - Through Data Transfer Module (DTM), sends a response when a MAPPER run is initiated by a QSNR statement (from another MAPPER run) or a CALL 'QSNR' (from a batch COBOL program or COBOL TIP transaction).
- Start (STR) - Starts a batch job.
- Submit SQL Statement (SQL) - Submits SQL syntax to a relational database manager.
- Trace (TRC) - Saves the SQL syntax generated by MAPPER Relational Interface (MRI) run statements in a specified report.
- UNIX Interface (UNX) - Accesses the UNIX operating system.

ART (Arithmetic)

The Arithmetic (ART) statement performs arithmetic operations on variables or constants. Variables capture the resulting numbers. Use the ART statement for complex operations; use a Change Variable (CHG) statement for simple computations.

In ART statements, you can use A, F, and I type variables containing numeric characters.

Format

@ART ***exp vrslts*** **.**

Following is a description of the fields:

Field	Description
exp	Arithmetic expression or expressions.
vrslts	Variables to capture the results of the expressions. Initialize the variables for the number of results you want to capture.

Operators

Table 7-2 lists arithmetic operators that you can combine to form an expression.

Table 7-2. ART: Arithmetic Operators

Operator	Operation	Expression	Description
+	Addition	a+b	Value a plus value b
-	Subtraction	a-b	Value a minus value b
/	Division	a/b	Value a divided by value b
//	Integer Division	a//b	Value a integer divided by value b
*	Multiplication	a*b	Value a times value b
**	Exponentiation	a**b	Value of a raised to the power of value b
-	Unary Minus	-a	Negative the value of a

Note: Values a and b are real integers and numbers and can include decimal fractions or expressions composed of such numbers.

Formulating Arithmetic Expressions

When formulating expressions, consider the following:

- Specify arithmetic operators for every operation; for example, enter the operation *a* times *b* as **a*b**. Forms such as (a)(b) or ab are not valid.
- Do not precede or follow operators with spaces.

Table 7-3 shows the priority by which the calculator of the MAPPER system performs arithmetic operations. For operators of equivalent priority, the operators are processed from left to right unless you change the priority with parentheses (see "Changing the Hierarchy of Expressions" in this subsection).

Table 7-3. ART: Priority of Arithmetic Operations

Priority	Operator	Operation
First	-	Unary minus
Second	**	Exponentiation
Third	*,/,//	Multiplication, division, integer division
Fourth	+,-	Addition, Subtraction

Example

Raise 3 to the 4th power, divide the result into 2, and place the answer in v1:

```
@art 3**4/2 v1f6.2 .
```

Multiple Expressions

Evaluating multiple expressions in a single statement is more efficient than using a separate ART statement for each expression.

Add v33 to v34 and put the answer in v36i3; subtract v34 from v33 and put the answer in v37i3 (note that a semicolon [;] separates the expressions):

```
@art v33+v34;v33-v34 v36i3,v37i3 .
```

Internal Computation

You can refer to variables that are created internally by an ART statement (a, b, and so on), then use these variables for computing expressions within the same ART statement, as in this example:

```
@art v1+v2;a*v3;b+5 ,v4i3,v5i3 .
```

In this example, the addition of v1 to v2 produces answer a, which is used in the second expression. The second expression, a*v3, produces answer b, which is used in the third expression. The first subfield in the variables field is skipped and a comma is used in its place, so the answer from the first expression (a) is not captured in a variable. V4 contains the answer to the second expression, and v5 contains the answer to the third expression.

Negative Numbers

If it is possible that a variable used in an arithmetic expression has a negative number, place the variable in parentheses; otherwise, the calculator reads it as part of an expression and the run errs. Place all negative numbers in arithmetic expressions in parentheses. For example, in this statement, v5 contains a negative number:

```
@art 3+(v5) v6i3 .
```

Changing the Hierarchy of Expressions

Use parentheses to change the hierarchy of expressions. In this example, 2 is divided by 3, the product is raised to the power of 4, and the answer is placed in v1:

```
@art (2/3)**4 v1f6.2 .
```

Arithmetic and Trigonometric Functions

You can perform the arithmetic and trigonometric functions shown in Table 7-4 using an ART statement. Note that *x* is a numeric value (whole or fraction) or an arithmetic expression.

Table 7-4. ART: Arithmetic and Trigonometric Functions

Function	Description
ABS(*x*)	Absolute value or magnitude of *x*
ACOS(*x*)	Arc cosine: angle in radians that has a cosine of *x*
ASIN(*x*)	Arc sine: angle in radians that has a sine of *x*
ATAN(*x*)	Arc tangent: angle in radians that has a tangent of *x*
CBRT(*x*)	Cube root of *x*
COS(*x*)	Cosine of *x* radians
CTN(*x*)	Cotangent of *x* radians
DEG(*x*)	*x* radians expressed in degrees
EXP(*x*)	Exponent: natural number e raised to power *x*
FRAC(*x*)	Fractional portion of *x*
HCOS(*x*)	Hyperbolic cosine of *x*
HSIN(*x*)	Hyperbolic sine of *x*
HTAN(*x*)	Hyperbolic tangent of *x*
INT(*x*)	Integer portion of *x*
LOG(*x*)	Logarithm of *x* in base "e"
LOG10(*x*)	Logarithm of *x* in base 10
PI	Pi (π)
RAD(*x*)	*x* degrees in radians
SIN(*x*)	Sine of *x* radians
SQRT(*x*)	Square root of *x*
TAN(*x*)	Tangent of *x* radians

ASR (A Series Run)

This section applies only to the A Series MAPPER System.

The A Series Run (ASR) statement enables you to execute A Series programs.

Format

```
@ASR[,rmt?,xmt?,ci,di,ri,hi?,in,co,do,ho?,out,lab] fn [args] .
```

Following is a description of the fields and subfields:

Field	Description
rmt?	Program uses a remote file, Y or N. Default = N (remote file is not allowed).
xmt?	After program executes, require user to press **Transmit** before returning to MAPPER software, Y or N. Default = N (user automatically returns to MAPPER software).
ci,di,ri	Report to send to the program as input (must be specified if *in* is specified).
hi?	Append headings to the input file, Y or N. Default = N.
in	INTNAME of the input file of the program run. If specified, the following filecard is added to the program: FILE *in-int* (KIND=DISK,TITLE=*fn-inputrpt*,DEPENDENTSPECS=TRUE); *in-int* is the INTNAME specified in the *in* subfield. *fn-inputrpt* is the file name (supplied by MAPPER software) containing the input report specified in the *ci,di,ri* subfields.
co,do	Cabinet and drawer in which to place the result (must be specified if *out* is specified).
ho?	Append headings to the output file when it returns to the MAPPER system, Y or N. Default = Y.

continued

continued

Field	Description
out	INTNAME of the output file returned to the MAPPER system. If specified, the following filecard is added to the program: **FILE *out-int* (KIND=DISK,TITLE=*fn-outputrpt*,MAXRECSIZE=*line*);** ***out-int*** is the INTNAME specified in the ***out*** subfield. ***fn-outputrpt*** is a file to return to the MAPPER system. ***line*** is the line size of the drawer specified in the ***co,do*** subfields.
lab	Label to go to if an error occurs.
fn	Name of the object code file to execute.
args	Arguments passed to the program, if the program takes arguments (maximum of 255 characters). Do not specify ***args*** if the program does not take arguments.

Comments

- If the file name contains spaces or if you specify arguments, enclose the entire file name and the arguments within apostrophes ('); however, do not enclose variables, such as v1 or v2, within apostrophes if they are used as arguments. For example, `'*OBJECT/ABC ON PACK'` or `'*OBJECT/ABC 1,CDE,'v2`.
- The input report, if specified, is file-equated to the INTNAME specified in the ***in*** subfield. For output, the file whose INTNAME is specified in the ***out*** subfield is file-equated to a temporary file of the appropriate size for the drawer.
- The ***in***, ***out***, ***fn***, and ***args*** fields are case sensitive. The file name ABC cannot be accessed as abc.

Examples

Run the program (NEWUSER)OBJECT/GEN/SALES/REPORT ON PACK using a remote file, create a temporary file, which is file-equated to a file with an INTNAME of OUTPUT, and return it as a result in cabinet 0, drawer B:

```
@asr,y,,,,,,,0,b,,\
OUTPUT '(NEWUSER)OBJECT/GEN/SALES/REPORT ON PACK' .
```

Run the program *OBJECT/PROD_STATUS using a remote file and the arguments ADD and v3:

```
@asr,y,n '*OBJECT/PROD_STATUS ADD,'v3 .
```

AUX (Auxiliary)

This section does not apply to the BTOS II MAPPER System.

The Auxiliary (AUX) statement sends a report or result to an auxiliary printer connected to a display terminal.

1100: On OS 1100 MAPPER Systems, you can also send reports or results to other kinds of auxiliary devices, such as diskette devices.

Format

```
@AUX,c,d,r,sn,[dev,dlnos?,f,tabs?,dhdgs?,d1char?,lsp,transp?,
unit,sl,spcc] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report to print.
sn	Station number where the auxiliary printer is connected.
dev	Auxiliary printer device name. Default = COP. To specify a device other than the default, see the coordinator. **1100:** On OS 1100 MAPPER Systems, you can specify devices other than printers. Also, this subfield is not optional; there is no default.
dlnos?	Delete the line numbers, Y or N. Default = N. Use Y if the report is 132 characters and you are printing in basic format.
f	Report format to be printed. Default = basic format. **1100:** On OS 1100 MAPPER Systems, default = basic format or, if a report was on display, the fields currently displayed by means of the VIEW function or those selected by an FMT or SFC run statement.
1100: *tabs?*	Include tab characters in the output, Y or N. Default = N, change them to spaces.
dhdgs?	Delete the report headings and end report line in the output, Y or N. Default = N.
d1char?	Delete the first character of each line in the output, Y or N. Default = N.
lsp	Line spacing (1, 2, or 3). Default = 1.

continued

continued

Field		Description
1100:	*transp?*	Print reports exceeding 132 characters, write lines exceeding 80 characters (for cassettes and diskettes), Y or N. Default = N.
1100:	*unit*	Unit on which to locate data in subsequent search operations. (This subfield is applicable only for cassettes and diskettes.)
sl		Site letter of the station to which the printer is attached. Default = local site. **1100:** This subfield is not available on OS 1100 MAPPER Systems.
spcc		Special print control character used to specify printer device controls. Default = blank, printer ignores control characters in text and prints them literally. The usual printer device control code is a tilde (~). To change the control character, see the *Manual Functions Reference*.

Comments

- An AUX statement does not create a result.
- You can direct output from an AUX statement to configured stations only.

Note: *For printing to start, the sign-on screen must be displayed on the terminal connected to the specified printer.*

- For more information on printing, such as suspending or stopping printing or emphasizing text, see the *Manual Functions Reference*.

Example

Print report 2B0, at station 9 on device name COP with deleted line sequence numbers, deleted headings, and double spacing:

```
@aux,0,b,2,9,cop,y,,,y,,2 .
```

BFN (Binary Find)

The Binary Find (BFN) statement finds items very quickly in sorted reports by sampling the data at midpoint in the report or series of reports to determine whether the data to be found precedes or follows this point (therefore the term binary). With the N or O option, it creates a result.

Format

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

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report to scan.
l	Line number where the binary find starts its first sample. Designating a line number slightly before the general location of the data speeds up the binary find process. If the data is actually located before the line number specified, the BFN statement searches the entire report to find the data. Use a Find (FND) statement if you want to locate data after a specific line.
lab	Label to go to if no finds are made or in case of an error.
o	Options field (see Options).
cc	Column-character positions or names of the fields to scan.
ltyp	Line type to scan (if you specify the A option, you can leave this subfield blank, but enter the comma).
p	The item to find. Additional find parameters: K Designates fields to compare (use with N, K, B, or @ option) = Designates result field (use with B or N option) =N Designates integer subtotal field (use with N option)
vrpt	Variable to capture the report number where the find was made.

continued

continued

Field	Description
vlno	Variable to capture the line number where the find was made. If no finds are made, the first variable contains the report number where a find should have been made; the second variable gives the line number preceding the line where a find should have been made. With the O option, four variables are produced: 1st Number of finds 2nd Number of lines in the result 3rd Report number of the first find 4th Line number of the first find

Options

A Processes all line types.

B Builds an index containing the first data line of each report. Use an index to speed up the find operations across multiple reports. Enter a K in the *p* (parameter) field for the target field or fields. Be sure the targets do not cross reports.

1100: On OS 1100 MAPPER Systems, targets may cross reports.

With the B option, the BFN statement creates a result that you must replace into the report immediately preceding the first report of the specified range.

If you would like the report numbers where the target items reside placed in a field, enter an equal sign in the corresponding parameter field.

If any reports in the range of reports are empty (headings only), you must add one blank line into the empty report before building the index. After the index is built, delete the blank line.

1100: On OS 1100 MAPPER Systems, empty reports may exist within the range of reports being processed.

C(S) Distinguishes between uppercase and lowercase letters.

1100: C(*x*) Alters the find process based on the character set order. Ordinarily the system processes the report based on the character set of the drawer. The C option allows you to choose the character set type on which to base the find. Use one of the following:

- C(F) Full character set (FCS)
- C(L) Limited character set (LCS)
- C(S) Strict comparison; distinguishes between uppercase and lowercase letters.

E Displays last item only if the target item appears more than once. Do not use this option with the O option.

F*n,n,...* Specifies the order of sorted fields when searching multiple fields. For *n*, type the corresponding sort parameter you used for the fields from left to right.

I[*n*] Uses the index in drawer at report *n*. Enter only I for the default report (report 2); the BFN statement scans report 2 or report *n* to determine where to find items, assuming that the reports having data to scan follow the index report.

K Verifies that reports are sorted in ascending order. Enter a K in the corresponding parameter field. Note that when you specify the K option, the BFN statement tests each line of every report specified, so use it with discretion on large databases.

N Creates a result containing the last occurrence of each different character string found in the specified field and the number of occurrences of each string. Do not use this option with the O option. See the *Manual Functions Reference* for details.

O	Creates a result containing the items found. (Also, you can capture the information in four variables instead of just two, as previously described.) Do not use this option with the E or N options.
P	Includes trailer lines (all line types other than the target line type) from the last valid find in the result (valid only with the N option).
Q	Quickly finds an item that appears only once in the report. Use the Q option if you know that the item appears only once in the report. When used with the O option, no trailer lines are included.
R*x*[-*y*]	Scans a range of reports from report number *x* to report *y*. Normally, when you process all reports in a drawer, report numbers 1 and 2 are skipped. You can use the R option to specify a range of reports, including reports 1 and 2. Note that you must specify a range; it cannot be a series of selected reports such as R3,5-10. All reports in the range must exist.
S	Scans each report separately (data does not have to be sorted across all reports).
U	Sets an update lock on the report in which the find is made (or would have been made).
@	Finds the first blank line or fields within a report (use to find a place to enter data). Use this option in runs that write blank lines at the end of the report. When a blank field is found, it is the first blank field after the last line with data in it. If lines with blank fields are interspersed with lines containing data, the blank fields are not found. When using the @ option, enter a K in the corresponding parameter field. Finds a slant as data.

Reserved Word

STAT1$ contains one of the following codes in case of an error:

1 Data not found
2 All lines with space fields filled (@ option)
3 Data not sorted

Comments

- If you do not specify an N or O option, you must examine the variables to determine where the find was made.
- Before executing the BFN statement, be sure the data is sorted (*across* reports, if processing a range of reports) on the fields or partial fields to search.
- If the BFN statement detects a sort order discrepancy, does not find data, or does not find blanks when looking for them, it gives control to the label specified in the statement.
- If a find is made, the report in which the find is made becomes the current -0. Any previous -0 result is released.
- If you specify a range of reports, make sure there are no empty reports (containing no valid data or headings only) within the range. If the BFN statement encounters an empty report, then finds the data that matches the search criteria in a subsequent report, it ignores any reports preceding the empty report. In this case, no system message is supplied because a valid find is made.

1100: On OS 1100 MAPPER Systems, empty reports may exist within the range of reports being processed.

Example 1: Finding All Occurrences of an Item

Find all occurrences of blackbox4 in report 1C0:

```
@bfn,0,c,1 o 'producttype' □,blackbox4 .
```

where:

`0,c,1`	Searches report 1C0
`o`	Creates a result (O option)
`'producttype'`	Searches the Product Type field
`□`	Processes tab lines only
`blackbox4`	Finds blackbox4s

Example 2: Finding the Only Occurrence of an Item

Use the Q option to quickly find the only blackbox4, and capture the line number where the find is made in <line>:

```
@bfn,0,c,1,,099 q 2-9 □,blackbox4 ,<line>i3 .
```

where:

`0,c,1`	Searches report 1C0
`099`	Goes to label 99 in case of no finds or an error
`q`	Quickly finds the item (Q option)
`2-9`	Looks in the Product Type field (column 2 for 9 characters)
`□`	Processes tab lines only
`blackbox4`	Finds the only blackbox4
`<line>`	Captures the line number where the find is made in variable <line>

Using the IBFN Run to Create a BFN Statement

This section on using the IBFN run applies only to the OS 1100 MAPPER System.

You can use the Iterative Binary Find (IBFN) run to create a BFN statement. For information on how to use the iterative runs, see the *Manual Functions Reference*.

BR (Background Run)

This section does not apply to the OS 1100 MAPPER System. See "BR (Background Run): OS 1100" in this section for a description of the BR statement as it applies to the OS 1100 MAPPER System.

The Background Run (BR) statement starts a MAPPER run at your site and executes it in background, allowing you to continue executing other functions and runs at your terminal.

The system removes the current (-0) result from the run, if one exists, passes it to the background run, and starts the run in background. The run containing the BR statement then continues at the next statement.

Format

```
@BR[,c,d,r] run[,vld] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report to pass as a -0 result to the background run. Default = current -0 (the system removes the current result from your run, if one exists, and passes it to the background run).
run	Name of the run.
vld	Variables, literals, reserved words, constants, or any combination of these, to be picked up by the background run via INPUT$. You can pass up to 40 variables; however, the maximum total number of characters for the *run* and *vld* fields together is only 200 characters.

Reserved Word

ORSTAN$ contains the station number that started the background run (contains zero if the run is not being executed in background).

Comments

- You cannot use the following statements in a background run: DSG, DSM, DSP, DSX, GOC (when display requested), GS (when display requested), HST, ITV, OTV, OUM, OUT, OUV, RDB, REL, SC, XIT, or XUN statements in a background run.

  Note: *Since this list encompasses several different kinds of MAPPER systems, some run statements may not be available on your system.*

- A Return Remote (RTN) statement in a background run sends the specified report to the calling terminal.
- If the background run fails, the run user receives a message indicating there was an error.
- To check the status of the run, use the System function.
- To terminate an active background run, use the Kill function. To terminate a queued background run, contact the MAPPER system coordinator.

Example

Start the test run, pass the letters abcd as input to the run, and pass report 2B0 for processing in the run:

```
@br,0,b,2 test,abcd .
```

The run containing the BR statement continues at the next statement after the background run starts.

BR (Background Run): OS 1100

This section applies only to the OS 1100 MAPPER System. See "BR (Background Run)" in this section for a description of the BR statement as it applies to other MAPPER systems.

The Background Run (BR) statement starts a MAPPER run and executes it in background, allowing you to continue executing other functions and runs at your terminal.

The system removes the current (-0) result from the run, if one exists, passes it to the background run, and starts the run in background. The run containing the BR statement then continues at the next statement.

Format

```
@BR[,sn,lab] run[,vld] .
```

Following is a description of the subfields:

Field	Description
sn	Station number to be notified when the background run completes. Default = no station notified.
lab	Label to go to if the number of active background runs has already reached the maximum allowed at your site. Default = run containing the BR statement stalls until the background run is able to start.
run	Name of the run.
vld	Variables, literals, reserved words, constants, or any combination of these, to be picked up by the background run via INPUT$. You can pass up to 40 variables; however, the maximum total number of characters for the *run* and *vld* fields together is only 640 characters.

Reserved Word

ORSTAN$ contains the station number from the *sn* subfield of the BR statement that started the run, or the originating station number if not supplied in the BR statement (contains zero if the run is not being executed in background).

Comments

- Have the coordinator register the run in one of the following ways:
  - As a normal run eligible for background execution (you can execute the run either normally or in background)
  - As a background run only
- You cannot use the following statements in a background run: DSG, DSM, DSP, DSX, GOC (when display requested), GS (when display requested), ITV, OUM, OUT, OUV, REL, RSI, SC, XIT .
- You can use the DSG, OUT, and SC statements if you send the output to another terminal.
- In addition to the statements listed above, the background run must not contain a BR statement. In other words, a background run may not start another background run.
- If the background run fails, the run user receives a message indicating there was an error.
- To check the status of the run, use the Run Status (RS) function.
- To terminate an active background run, use the Stop function.

Example

Start a background run and transfer data to it:

```
@br,123,099 test,abcd
```

where:

123	Notify station 123 when the background run completes.
099	Go to label 99 if the number of active background runs has already reached the maximum at this site.
test	Start the run named test.
abcd	Pass the literal value abcd as input to the run.

The run containing the BR statement continues at the next statement after the background run has started.

BRK (Break)

The Break (BRK) statement turns the output area of a run into the current -0 result and creates a new output area (containing only a date line).

Format

```
@BRK[,c,d,q] .
```

Following is a description of the subfields:

Field	Description
c,d	Cabinet and drawer letter of the new output area. The drawer letter is required if a cabinet is specified. Default = cabinet and drawer of the run control report or cabinet and drawer specified on a previous BRK statement.
 **1100:** *q*	Estimated number of output lines. For output exceeding 500 lines, estimating improves efficiency by reducing I/Os.

Comment

- When a run encounters a BRK statement, all data currently in the output area becomes the current result, -0 and a new, empty output area (containing only a date line) is created. If you specify a cabinet and drawer, the location of the current result is not affected. However, the new output area is placed into the specified cabinet and drawer.

Examples

This example uses multiple BRK statements:

```
1. @brk .
       Data1
2. @brk,0,b .
       Data2
3. @brk .
       Data3
4. @brk .
```

Statement 1. The current output area resides in the same cabinet and drawer as the run control report.

Statement 2. Following this statement, `Data1` is the current -0 which is in the same cabinet and drawer as the run control report. The new output area however, resides in cabinet 0, drawer B.

Statement 3. Following this statement, `Data2` is in the -0 result in cabinet 0, drawer B.

Statement 4. Following this statement, `Data3` is in the -0 result in cabinet 0, drawer B.

1100: This example applies only to OS 1100 MAPPER Systems.

This example places the output area into a result and estimates that the following output area will be 2,500 lines:

```
@brk,,,2500 .
```

CAL (Calculate)

The Calculate (CAL) statement performs arithmetic calculations and conditional evaluations on reports or results, creating a result (except with the O option). See the *Manual Functions Reference* for details on the CAL function.

To create an update result that allows you to blend the changed lines back into the original report or delete lines from the original report, use the Calculate Update (CAU) statement. The CAU statement uses the same fields and subfields as the CAL statement. See the online help system (HELP,@CAU) for more information.

Format

```
@CAL,c,d,r[,l,q,lab] o cc ltyp,p eq [vrslts] .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report to process.
l	Line number at which to start processing.
q	Number of lines to process.
lab	Label to go to if no data exists. (Be sure to use this field if there is any possibility that the report does not contain data; otherwise, no result is created.)
o	Options field (see Options).
cc	Column-character positions or names of the fields to process.
ltyp	Line type to process. If you specify the A option, you can leave this subfield blank, but enter the comma.
p	Parameters: Alphabetic field labels Vertical operators (+,/,<, and >)

continued

continued

Field	Description
eq	Equations in the format: ***receiving-label*[,*options*]=*expression*[;...]** ***receiving-label*** can have alphabetic field labels, or one- to six-alphabetic character value labels including the characters $, %, !, and ?. (It can also have constant label LT.) Note that ***receiving-label*** can be a partial label in the format: ***receiving-label*(*x*-*y*)** *x* is the starting column and *y* is the number of characters. ***options*** are equation options E, I, J(*x*), K*n*, N*n*, R*n*, V, X, and *. Use them to specify options on individual equations. To remove a function option from an equation, enter a minus sign (-) after the equation option. For example, J-, K-, V-, and so on. ***expression*** is a combination of one or more operands and zero or more operators regarded as a single numeric value. Separate multiple equations with a semicolon (;).
vrslts	Variables to capture the results of vertical operations and the contents of value labels in the order initialized. Variables that capture vertical operations are loaded in the order their fields exist in the report.

Options

A	Processes all line types.
C	Conditionally displays specific result lines. After all equations are processed, include only those lines that meet a true condition based on the last If Conditional (IF) statement in the result. For example, if the last IF statement is IF:A=0, include only lines where field A is equal to zero in the result.
E	Erases fields (fill them with spaces) if the answer equals zero.
I	Produces integer results, truncating any fractional part (numbers to the right of a decimal point) in the result.

J(*x*) Justifies the numeric result value, where *x* is one of the following justification options:

- C Inserts commas in the integer portion every three digits, eliminating nonsignificant zeros. Places the resulting value in the leftmost portion of the field.
- L Left-justifies and eliminates nonsignificant zeros. Places the resulting value in the leftmost portion of the field.
- R Right-justifies and eliminates nonsignificant zeros. Places the resulting value in the rightmost portion of the field.
- X Eliminates leading nonsignificant zeros. Places the resulting value in the leftmost column and fill the remaining columns with zeros.
- Z Eliminates nonsignificant zeros, right-justify, and fills the preceding columns with zeros.

K*n* Initializes a value label to *n*. Default value = 0.

L Lists all value label names and their final values at the end of the result.

N*n* Substitutes the numeric value of *n* for nonnumeric fields. The default value of nonnumeric fields is zero. A nonnumeric field has either no data (all space or tab characters) or data that has a nonnumeric character in its leftmost significant position.

O Omits result. Only receiving variables are loaded.

R*n* Rounds results to the nearest *n*: R.0000000000000001 through R100000 (nearest 100,000 units).

Note: *To control rounding by equation, use R as an equation option.*

S(s) Distinguishes between uppercase and lowercase characters when processing character strings.

1100: S(x) Defines character set interpretation as x (F, L, or S). Note that the system compares limited character set (LCS) strings to LCS internal codes, full character set (FCS) strings to FCS internal codes, and interprets uppercase and lowercase characters the same.

- F Use FCS internal codes. Use only when processing LCS reports.
- L Use LCS internal codes. Use only when processing FCS or full character set upper (FCSU) reports.
- S Use strict character set internal codes to differentiate between uppercase and lowercase characters. Only with FCS reports.

T Includes both processed and unprocessed lines in the result. Do not use if you want to include only the line type being processed.

V Processes only those equations whose result values are calculated from valid data. Equations containing invalid data are not included in the result, and the receiving label is not altered. See the note following this options list.

X Excludes invalid values in minimum, maximum, sum, and average computations (MIN, MAX, SUM, AVG, VMIN, VMAX, VSUM, VAVG), as well as in functions specified by vertical operators. See the note following this options list.

* Flags all invalid results with an asterisk following the value. See the note following this options list.

Note: *Invalid data includes field labels representing nonnumeric data and value labels whose value was calculated from nonnumeric data or an invalid date. Nonnumeric fields are fields containing either nonnumeric data in the leftmost column of the field or no data.*

Reserved Words

STAT1$ contains the number of lines processed (of the line type specified).

STAT2$ contains the total number of lines in the report (excluding headings).

Priority of Operations

Table 7-5 shows the priority by which the calculator performs arithmetic operations.

Table 7-5. CAL: Priority of Arithmetic Operations

Priority	Operator	Operation
First	-	Unary Minus
Second	**	Exponentiation
Third	*,/,//	Multiplication, Division, Integer Division
Fourth	+,-	Addition, Subtraction

Table 7-6 shows the priority by which the MAPPER system evaluates relational operations.

Table 7-6. CAL: Priority of Relational Operations

Priority	Operator	Relational Operation
First	=	Compare equal to
	<> or ><	Compare not equal to
	>	Compare greater than
	<= or =<	Compare not greater than (less than or equal to)
	<	Compare less than
	>= or =>	Compare not less than (greater than or equal to)
Second	&	AND (Boolean)
Third	,	OR (Boolean)

Note: *The operators ampersand and comma (& and ,) do not perform a numeric comparison but are based on a true/false concept. See the* Manual Functions Reference *for details on using relational and logical operators.*

True/False Conditions

Table 7-7 shows the result of all possible true/false conditions. *value1* and *value2* are usually relational expressions (such as a>1000).

Table 7-7. CAL: AND and OR True/False Conditions

AND Operation			OR Operation		
value1 & *value2*		Result	*value1* , *value2*		Result
False	False	False	False	False	False
True	False	False	True	False	True
False	True	False	False	True	True
True	True	True	True	True	True

Comments

- You can use the following internal arithmetic and trigonometric functions in the equations:

ABS(*x*)	LOG(*x*)
ACOS(*x*)	LOG10(*x*)
ASIN(*x*)	MAX(*x1,...,xn*)
ATAN(*x*)	MIN(*x1,...,xn*)
AVG(*x1,...,xn*)	MOD(*x,y*)
CBRT(*x*)	RAD(*x*)
COS(*x*)	RAN(*x,y*)
CTN(*x*)	SIN(*x*)
DEG(*x*)	SQRT(*x*)
EXP(*x*)	SUM(*x1,...,xn*)
FRAC(*x*)	TAN(*x*)
HCOS(*x*)	VAVG(*x1,...,xn*)
HSIN(*x*)	VMAX(*x1,...,xn*)
HTAN(*x*)	VMIN(*x1,...,xn*)
INT(*x*)	VSUM(*x1,...,xn*)

See Arithmetic (ART) for an explanation of these functions.

- You can use variables in any field of the CAL statement, including equations. Variables can contain entire equations or parts of equations.

  Since the system interprets variables enclosed in apostrophes as literal data (for example, the statement uses literal v1 rather than the contents of the variable v1), do not enclose variables in apostrophes.

- Use a DEF statement to produce a numeric value that defines the contents of a report field. Use the following format:

  DEF(*field-label*)

  Following are the report field values and what they represent:

Value	Contents of Report Field
0	All tab characters or spaces or both
1	All numeric characters
2	All alphabetic characters
3	Alphabetic and numeric characters
4	All special characters
5	Special and numeric characters
6	Special and alphabetic characters
7	Special, numeric, and alphabetic characters

- You can include both characters and spaces in the contents of a report field.

- Use the any of the following conditional statements in the equations:

  IF:*expression*[; . . .]

  THEN:*equation*[; . . .]

  ELSE:*equation*[; . . .]

  FIRST:*equation*[; . . .]

  SKIP:*expression*[; . . .]

  EXIT:*expression*[; . . .]

  For details on conditional statements, see the *Manual Functions Reference*.

Date and Time Processing

You can use the CAL statement to perform computations on dates and times. The CAL statement converts the date and time data to numeric values and processes them with equations. See the *Manual Functions Reference* for details on using the CAL function to process dates and times.

Character String Processing

You can use the CAL statement to manipulate character strings in reports.

Enclose all literal values in quotation marks (").

1100: On OS 1100 MAPPER Systems, to use literal data that contains spaces, enclose the data in apostrophes or quotation marks.

Since quotation marks are not allowed in the limited character set (LCS), you must use the reserved word TIC$ or a variable containing an apostrophe before and after literal data in runs residing in LCS reports.

See the *Manual Functions Reference* for details on processing character strings.

Using the ICAL Run to Create a CAL Statement

You can use the Iterative Calculate (ICAL) run to create a CAL statement. For information on how to use the ICAL run, see the online help system (HELP,ICAL).

CAL Statement Examples

Note: All CAL examples except the last one process tab lines in report 1C0.

Example 1: Multiplying Two Fields

Multiply the Space Req field by the Demo Quantity field:

```
@cal,0,c,1 '' 'spacereq','demoquan',\
'demoresults' □,a,b,c c=a*b .
```

where:

`''`	Uses no options
`'spacereq'` `a`	Labels the Space Req field: Field a
`'demoquan'` `b`	Labels the Demo Quantity field: Field b
`'demoresults'` `c`	Labels the Demo Results field: Field c
`c=a*b`	Multiplies the Space Req field (a) by the Demo Quantity field (b) and places the product in the Demo Results field (c)

Example 2: Moving Values into Fields Based on Values

Move all values that exceed 24,000 from one field into another field:

```
@cal,0,c,1 '' 'retail $$$$','demoresults' \
□,a,b if:a>24000;then:b=a .
```

where:

`''`	Uses no options
`'retail $$$$'` `a`	Labels the Retail $$$$ field: Field a
`'demoresults'` `b`	Labels the Demo Results field: Field b
`if:a>24000;` `then:b=a`	Puts all Retail $$$$ field (a) values that exceed 24,000 into the Demo Results field (b)

Example 3: Averaging Items in a Field

Calculate the vertical average of the Sales Commis field:

```
@cal,0,c,1 r.01 42-7,65-15 □,a,b b=vavg(a) .
```

where:

`r.01`	Uses the R option, rounds result to nearest one hundredth
`42-7`	Labels the Sales Commis field (column 42 for seven characters):
`a`	Field a
`65-15`	Labels the Demo Results field (column 65 for 15 characters):
`b`	Field b
`b=vavg(a)`	Puts the cumulative vertical average of the Sales Commis field (a) into the Demo Results field (b)

Example 4: Testing Literal Data

The following example uses quotation marks when testing a literal value:

```
@cal,0,c,1,,,099 c 2-8 □,a if:a="blackbox" .
```

where:

`099`	Goes to label 99 if no data exists
`c`	Uses the C option to include in the result only those lines that meet a true condition based on the last IF statement
`2-8`	Labels the first eight characters of the Product Type field (column 2 for 8 characters):
`a`	Field a
`if:a="blackbox"`	Includes all lines with the word blackbox in the first eight characters of the Product Type field

Example 5: Calculating Vertical Totals and Maximum Values

Calculate the vertical totals of the Retail $$$$ field and the Demo Results field, and capture the highest value found in the Retail $$$$ field and the Whole Sale$ field:

```
@cal,0,c,1,,,099 l 25-7,33-8,65-15 □,a,b+,c+ \
maxa=vmax(a);maxb=vmax(b);c=b-a v1i9,v2i9,\
v3i9,v4i9 .
```

where:

`099`	Goes to label 99 if no data exists
`l`	Uses the L option to list all value label names and their final values at the end of the result
`25-7,33-8,65-15`	Labels the Whole Sale$ field (column 25 for 7 characters), the Retail $$$$ field (column 33 for 8 characters), and the Demo Results field (column 65 for 15 characters):
`a,b+,c+`	Fields a, b, and c, with the vertical operator + on fields b and c
`maxa=vmax(a);`	Puts the highest value found in the Whole Sale$ field (a) in the value label maxa
`maxb=vmax(b);`	Puts the highest value found in the Retail $$$$ field (b) in the value label maxb
`c=b-a`	Subtracts field a from field b and puts the difference in field c
`v1i9,v2i9, v3i9,v4i9`	Captures the vertical total of Retail $$$$ (field b) in v1, the vertical total of Demo Results (field c) in v2, the contents of the MAXA value label in v3, and the contents of the MAXB value label in v4

CALL (Call Subroutine)

The Call Subroutine (CALL) statement saves 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,...]) .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report containing an external subroutine.
lab	Label where subroutine begins. At the subroutine label, you must include the parentheses and a corresponding variable for each variable passed on the CALL statement. Use the following format for the label: *@lab*:(*v,v,...*)

continued

continued

Field	Description
(*v,v...*)	Variables to pass to the subroutine (maximum of 40). Parentheses are required (even if you do not pass any variables).
 **1100:** (*p,p...*)	Parameters to pass to the subroutine (maximum of 40). Parentheses are required (even if you do not pass any parameters). Enter one of the following kinds of parameters: *v* — Variable (system returns the new value from the subroutine). **v* — Variable (system retains the original value from the calling run). *'string'* — Literal character string, up to 256 characters. Enclose each string in apostrophes ('). To pass an apostrophe, use two apostrophes (''). At the label, enter variables for the strings. The system initializes these variables as S type variables of the same size as the strings.

Comments

- The statement returns only the -0 result created in the subroutine and the variables that you passed with the CALL statement. Renamed results (-1 to -8) remain unchanged.
- Use the RETURN statement to exit the subroutine and return to the line following the CALL statement.
- To pass all the variables in an array, specify only the name of the array; to pass a specific array member, specify the array name and member number ([*n*]). An entire variable array counts as one variable of the 40 allowed. See LDA in this section.

- Variables you pass with the CALL statement have a one-to-one relationship with the variables specified at the label. MAPPER software equates these variables by their position in the statement, not by their name. That is, the first variable in parentheses is equated with the first variable of the CALL statement, and so on.

  If the subroutine modifies the value of a variable in parentheses, the system passes the new value to the equivalent variable in the CALL statement.

- Any changes you make to variables you passed to the subroutine using the CALL statement still exist when control returns. Variables you did not pass to the subroutine remain unchanged when control returns to the calling run.

- Within the subroutine, you can use up to 199 variables (or the number configured for your site, up to 399), less the number of variables previously defined. Since you can pass up to 40 variables with the CALL statement, you can initialize and use up to 159 (or up to 359) additional variables within that subroutine, less the number previously defined.

1100: On OS 1100 MAPPER Systems, you can use the system limit of variables both in the calling run and in the subroutine.

When control returns to the calling run, the system returns only the variables specified on the CALL statement. The system releases all variables initialized within the subroutine.

- You can nest up to ten CALL subroutine levels (internal or external). The system keeps track of up to ten sets (levels) of variables. Each time you issue a CALL or Link to Another Run (LNK) statement, the system saves a level.

1100: On OS 1100 Systems, the system saves a level each time you issue a CALL statement, not a LNK statement.

- If you pass a string variable in a CALL statement, you cannot modify its size.

1100: On OS 1100 Systems, you can modify the size of string variables.

- The total number of variables used in all CALL statements cannot exceed 199 (or the number configured for the run).

1100: On OS 1100 Systems, you can use the system limit of variables in each of the CALL statements.

- Error routines registered in called routines override previously registered error routines; however, when control returns to the calling run, the system automatically reregisters the original error routine.

1100: On OS 1100 Systems, error routines registered in called routines do not override previously registered error routines.

- If you do not register an error routine in the called routine, in case of an error, control goes to the error routine in the calling run.

1100: On OS 1100 Systems, if the run fails, the system cancels all active subroutines.

- Do not place other run statements on the same line after the CALL statement. Other run statements following it on the same line are ignored. Put the next run statement on a new line.

Example 1

This statement transfers control to label 054 and passes only v102 and v103 to the subroutine. The subroutine uses the values of v102 and v103 for <drw> and <line>, respectively.

```
@ldv v101i4=1234,v102h1=b,v103i4=5 .
@call 054 (v102,v103) .
    .
    . (other processing)
    .
@054:(<drw>,<line>) .
@fnd,0,<drw>,,<line> a 'cust' |,amco <rpt>i4,<line> .
@return .
```

In the example, when control returns, the variables used in the main routine and subroutine have been affected as follows:

- V101 still contains 1234 because it was not passed to the subroutine.
- V102 (<drw> in the subroutine) still contains the letter b because the subroutine did not change it.
- V103 (<line> in the subroutine) contains a new value because the Find (FND) statement in the subroutine changed it.
- <rpt> is not defined any longer because it was used only in the subroutine.

Example 2

This statement transfers control to label 99 and passes v102, v103, and v104. The subroutine uses the values of v102, v103, and v104 for <scan>, <item>, and <qty>, respectively.

```
@call 099 (v102,v103,v104) .
    .
    . (other processing)
    .
@099:(<scan>,<item>,<qty>) .
    .
    . (other processing using <scan1>, <item>, and <qty>)
    .
@srh,0,b dha 'product type' |,<item> <qty>,<scan> .
    .
    . (other processing)
    .
@return .
```

The system returns v102, v103, v104, and the -0 result created in the subroutine. All other variables remain unchanged since you did not pass them to the subroutine.

Example 3

1100: This example applies only to OS 1100 MAPPER Systems.

This statement transfers control to label 100 and passes v101 and v102. The asterisk (*) before v102 specifies that its value be retained upon return to the calling run, even if it is changed in the subroutine. The subroutine changes both variables.

```
@call 100 (v101,*v102) .
.
. (other processing)
.
@100:(<data1>,<data2>) .
@ldv <data1>=5000,<data2>=2500 .
@return .
```

The system returns 5000 in v101, but does not change v102 because it was preceded with an asterisk when passed with the CALL statement.

Example 4

1100: This example applies only to OS 1100 MAPPER Systems.

This statement transfers control to label 100 and passes v101 and strings hokey and pokey. The subroutine automatically initializes <data2> and <data3> as S type variables of the same size as the strings.

```
@call 100 (v101,'hokey','pokey') .
.
. (other processing)
.
@100:(<data1>,<data2>,<data3>) .
```

The system returns the current value of v101, but does not return any variable comparable to <data2> or <data3>.

CHD (Command Handler)

The Command Handler (CHD) statement registers a routine to be executed when the user of the run enters information in the control line after a Display Message (DSM), Display Report (DSP), Output (OUT), or Screen Control (SC) statement.

Note: *Do not use the CHD statement unless you are an advanced run writer. It is intended primarily for intercepting and interpreting commands that normally go to MAPPER software.*

Format

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

Following is a description of the field and subfields:

Field	Description
c,d,r	Report that contains a command handler routine. (Specify only if the routine is in another run control report.) **1100:** The report containing the command handler routine must be in the same character set type as the calling run control report.
rel?	Transfer release control (^) to the run, Y or N. Default = N. If you specify Y, the run user cannot execute the Release Display (^) function because the caret (^) is intercepted by the run.
lab	Label where the command handler routine starts. Use 0 to cancel the currently registered command handler routine. To begin the routine at the first line of the external run control report, use LIN1 in this field.

Reserved Words

ICVAR$ captures user input from the control line whenever the user transmits from the control line. Follow these guidelines when using ICVAR$:

- Use only with a CHD statement.
- Place ICVAR$ before the variable in the CHG statement, and put the CHG statement before the DSM, DSP, OUT, or SC statement.

CHG (Change Variable)

The Change Variable (CHG) statement initializes or redefines the contents of variables. See Section 4 for more information on initializing and redefining variables.

Format

```
@CHG v {exp|vld} .
```

or

```
@CHG rw v[,v...,v] .
```

Following is a description of the fields and subfields:

Field	Description
v	Variable you are changing. To define or redefine the variable, include the variable type and size.
exp	Expression by which you want to set the value of the variable.
vld	Value to place into the variable. You can specify the value using a variable, literal, constant, or reserved word.
rw	Reserved word that determines the kind of input data you want to capture in the variable.

Reserved Words

You can use only the following reserved words to capture user input with the CHG statement:

ICVAR$ (see CHD)
INMSV$ (see OUM)
INPUT$ (see OUT)
INSTR$ (see OUT)
INVAR$ (see OUT)
INVR1$ (see OUT)

Put reserved words that capture user input *before* the variable.

Examples

Set the value of <commission> with the solution of a simple expression:

```
@chg <commission> <sales> * <percent> .
```

Capture the current input in <name>, <address>, and <phone>:

```
@chg input$ <name>,<address>,<phone> .
```

Capture the next input entered on the control line in <controlln>:

```
@chg icvar$ <controlln> .
```

Using Expressions

You can use variables, constants, reserved words, and the following operators in a CHG statement expression:

- addition (+)
- subtraction (-)
- multiplication (*)
- division (/)
- integer division (//)

Precede each operator with a space. If you include a variable in the expression, the variable must have been defined prior to the CHG statement.

The system evaluates the expression from left to right, following no precedence rules. Therefore, use only simple expressions with the CHG statement.

Guidelines for Addition and Subtraction

Follow these guidelines when adding or subtracting with variables:

- A type variables:

  Produce numeric results whenever the variable contains numbers. Type A variables containing letters or special characters produce a result based on the order of the character set of the run control report (for example, A + 1 = B).

- H type variables:

  Produce a result based on the order of the character set, even for numbers (for example, 9 + 1 = : , not 10 [this assumes that the number 9 was right-justified in the variable]).

1100: On OS 1100 Systems, the example also assumes the run control report is in a full character set drawer.

- S type variables:

  Produce a result based on the order of the character set, even for numbers.

- 1100: O type variables:

  Produce octal answers.

- To simply increase or decrease the numeric value of a variable, use the Increment Variable (INC) or Decrement Variable (DEC) statement.

Processing Octal Variables

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

You can use the following operators with octal (type O) variables:

A Logical AND
O Logical OR
X Exclusive OR
L Left shift (also -)
R Right shift (also +)
C Circular shift

A shift of a type O variable is actually a bit-count shift. If the data being loaded is a literal value, MAPPER software assumes an octal value. If the data resides in another variable, MAPPER software assumes a decimal value and converts it to octal before processing.

Examples

Initialize <testoctal> as a four-character octal variable with an initial value of 2:

```
@chg <testoctal>o4 2 .
```

Perform a logical AND of the contents of <testoctal> with the number 7:

```
@chg <testoctal> <testoctal> a 7 .
```

Perform a shift on the contents of <testoctal> to the left one bit:

```
@chg <testoctal> <testoctal> L 1 .
```

CMP (Compare Report)

The Compare Report (CMP) statement compares the contents of a report or result with another report. The CMP statement performs a character-to-character comparison between the two reports. If any character differences are detected, the variables are loaded with their line numbers and column character positions and the run continues at the label (if specified).

Format

@CMP,*c1,d1,r1*,[*lin1*],*c2,d2,r2*[,*lin2,q,lab*] *o cc1 ltyp1,p1 cc2 Δltyp2,p2* [*vlno1,vcol1,vlno2,vcol2*] .

Following is a description of the fields and subfields:

Field	Description
c1,d1,r1	Report to compare.
lin1	Line number in the first report at which to begin the comparison. Default = line 2.
c2,d2,r2	Report to compare with the first report.
lin2	Line number in the second report at which to begin the comparison. Default = line 2.
q	Number of lines to compare. Default = all.
lab	Label to go to if the statement detects a difference.
o	Options field (see Options).
cc1	Column-character positions or names of the fields to compare in the first report.
ltyp1	Line type to compare in the first report. If you specify the A option, leave this subfield blank but enter the comma.
p1	Parameters for the first report.

continued

continued

Field	Description
cc2	Column-character positions or names of the fields to compare in the second report.
ltyp2	Line type to compare in the second report. If you specify the A option, leave this subfield blank but enter the comma.
p2	Parameters for the second report.
vlno1	Variable to capture the line number in the first report where the fields are not identical.
vcol1	Variable to capture the column in the first report where the fields are not identical.
vlno1	Variable to capture the line number in the second report where the fields are not identical.
vcol1	Variable to capture the column in the second report where the fields are not identical.

Options

A — Processes all line types. Each line is compared regardless of line type.

C(S) — Distinguishes between uppercase and lowercase letters.

1100: On OS 1100 MAPPER Systems, this option applies only to full character set (FCS) reports.

Examples

Compare reports 2D0 and 3D0:

```
@cmp,0,d,2,6,0,d,3,6,20,099 '' 12-9,61-17 *,1,2 \
12-9,61-17 *,1,2 v1i4,v2i3,v3i4,v4i3 .
```

where:

`0,d,2,6` `0,d,3,6,20`	Compares twenty lines, beginning at line 6, in reports 2D0 and 3D0
`099`	Goes to label 99 if the report fields are not the same
`''`	Uses no option
`12-9,61-17` `1,2`	Compares column 12 for 9 characters and column 61 for 17 characters in the issuing report
`12-9,61-17` `1,2`	with column 12 for 9 characters and column 61 for 17 characters in the receiving report
`*`	Compares asterisk lines only
`v1i4,v2i3,` `v3i4,v4i3`	If the report fields are not the same, captures issuing and receiving report line numbers and columns where the reports are not identical in v1, v2, v3, and v4

The run continues at the next run statement if the report fields are the same.

Compare reports 1C0 and 2C0:

```
@cmp,0,c,2,15,0,c,1,22,5,001 a 2-79 □,1 2-79 □,1 \
v1i4,v2i3,v3i4,v4i3 .
```

where:

0,c,2,15 **0,c,1,22,5**	Compares five lines, beginning at line 15 in report 2C0, and five lines, beginning at line 22 in report 1C0.
001	Goes to label 1 if the reports are not the same.
a	Compares all line types (line 15 compared to line 22 regardless of line type).
2-79	Compares column 2 for 79 characters in the issuing report.
2-79	with column 2 for 79 characters in the receiving report.
□	Processes tab lines (but A option overrides this subfield and compares all line types).
v1i4,v2i3, **v3i4,v4i3**	If the report fields are not the same, captures issuing and receiving report line numbers and columns where the reports are not identical in v1, v2, v3, and v4.

The run continues at the next run statement if the report fields are the same.

Compare reports 2B0 and 1C0:

```
@cmp,0,b,2,,0,c,1,,15 c(s) 15-9 □,1 2-9 □,1 \
v1i4,v2i3,v3i4,v4i3 .
```

where:

`0,b,2` `0,c,1,,15`	Compares fifteen lines, beginning at line 2 (default) in reports 2B0 and 1C0.
`c(s)`	Detects uppercase and lowercase differences.
`15-9` `2-9`	Compares column 15 for 9 characters in the issuing report with column 2 for 9 characters in the receiving report.
`□`	Compares tab lines only.
`v1i4,v2i3,` `v3i4,v4i3`	If the report fields are not the same, captures issuing and receiving report line numbers and columns where the reports are not identical in v1, v2, v3, and v4, then continues at the next statement.

Compare renamed results -1 and -2:

```
@cmp,-1,,-2,,,003 '' 25-6,39-5 □,1,2 5-6,110-5 \
□,1,2 v1i4,v2i3,v3i4,v4i3 .
```

where:

-1 **-2**	Compares all lines (default), beginning at line 2 (default), in renamed results -1 and -2.
003	Goes to label 3 if the reports are not the same.
''	Uses no option.
25-6,39-5 **1,2**	Compares column 25 for 6 characters and column 39 for 5 characters in the issuing report.
5-6,110-5 **1,2**	with column 5 for 6 characters and column 110 for 5 characters in the receiving report.
□	Compares tab lines only.
v1i4,v2i3, **v3i4,v4i3**	If the report fields are not the same, captures issuing and receiving report line numbers and columns where the reports are not identical in v1, v2, v3, and v4.

CNT (Count)

This section applies only to the OS 1100 MAPPER System.

The Count (CNT) statement computes subtotals, percentages, standard deviations, averages, entry counts, and other calculations on reports or results creating a result. See the *Manual Functions Reference* for more information.

Format

```
@CNT,c,d[,r] o cc ltyp,p [vrslts] .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report or drawer to process.
o	Options field (see Options).
cc	Column-character positions or names of the fields to process.
ltyp	Line type to process. If you specify the A option, you can leave this subfield blank, but enter the comma.
p	Parameters: 1-9 to specify key fields (at least 1 is required) N to specify numeric key fields (1N) D*f* to specify date key fields *f* = 0-8 to specify date format (See the S*n* option for date format codes.) Example: 1D0, specifies key field 1 in date format 0 T*f* to specify time key fields *f* = 0-3 to specify time format (See the S*n* option for time format codes.) Example: 2T3, specifies key field 2 in time format 3 R to set up report key fields Example: 1R, loads the field with the report number when processing a drawer or range of reports L to specify line key fields Example: 1L, loads the field with the input report line number
vrslts	Variable to capture the number of lines for the corresponding output result. Up to eight variables may be returned by the statement.

Options

A	Processes all line types.
B	Extends the boundaries for scaling intervals to the minimum and maximum values defined in the scaling option. Every interval between those boundaries is displayed, even those with no entries.
C	Distinguishes between uppercase and lowercase letters. Default = the system processes and displays the keys as uppercase.
D[*n*]	Processes only those keys that occur *n* or more times, where *n* is an integer greater than 1. Default = 2.
E	Extracts result fields from the last occurrence of each unique key. Default = first occurrence.
F	Extracts result fields from the first occurrence of each unique key. When using a minimum (<) or maximum (>) operator, default = minimum or maximum field.
H	Displays only the first set of headings in the result when you process more than one set of parameters in a single statement.
I	Includes out-of-range keys. Substitutes the minimum or maximum boundary value, when applicable, for an out-of-range key. The key field is then considered valid, and all its values are counted. Default = the line is discarded for values outside the boundaries.
N*n*	Substitutes invalid numeric data with the value *n*. Default = 0.
O	Omits data lines from the result. Only heading lines, grand total summary, and warning messages are included in the output result. The T and W options are automatically implied when the O option is specified.

P — Removes all non-referenced fields from the result and reorders the fields, from left to right, according to this hierarchy:

1. Key fields in numeric order
2. Extraction fields in alphabetical order
3. Computational fields (subtotals, averages, and so on)
4. Minimum fields
5. Maximum fields
6. Entry counts
7. Percentages

R*x-y* — Processes a range of reports from *x* through *y*.

R*x,y* — Processes reports *x* and *y*. The reports are processed in the order you specify.

R*x-y,z* — Processes reports *x* through *y* and also *z*. The reports are processed in the order you specify.

S*n* — Creates intervals or scales in key fields. Following are the formats for this option.

S*n*(*intv*[/*min*/*max*])	Scales numeric values.
S*n*D*f*(*intv*[/*min*/*max*/*strt*])	Scales dates.
S*n*T*f*(*intv*[/*min*/*max*/*strt*])	Scales times.

n — Number of the key field to scale (1 through 9).

f — Date or time format number.

Date format numbers.

0	YMMDD	5	DDMMYY
1	YYMMDD	6	MM/DD/YY
2	DD MMM YY	7	MONTH DD,YYYY
3	YDDD	8	MMDDYY
4	YYDDD		

Time format numbers.

0	HH:MM:SS	2	HHMMSS
1	HH:MM	3	HHMM

intv Interval size.

Date intervals:

*n*D	*n* = number of days (default)
*n*W	*n* = number of weeks
*n*M	*n* = number of months
*n*Y	*n* = number of years

Time intervals:

*n*S	*n* = number of seconds
*n*M	*n* = number of months
*n*H	*n* = number of hours

min Minimum value at which to begin scaling.

max Maximum value at which to end scaling.

strt Starting value.

If *intv* is:	Then *strt* is:
*n*D or *n*W	3-character day name, example: SUN
*n*M	3-character month name, example: MAY
*n*Y	2- or 4-digit year, example 90 or 1990
*n*S	same time format as data in the key field
*n*M	same time format as data in the key field
*n*H	same time format as data in the key field

T Displays total, overall summary values for each computed field at the end of each result. For example, subtotaling shows a grand total; averaging shows a grand average; entry counting shows total entries; minimum and maximum operators show the smallest or largest values in the field. You can also calculate the overall standard deviation of values in a field.

U Displays the upper limit of the interval for scaled results rather than the lower limit.

V	Does not count lines with invalid numeric, date, or time fields. Default = 0 for numeric operations; January 1, 1944, for date calculations; or 00:00:00 for time operations.
W	Adds messages at the end of the result showing the number of lines skipped due to invalid key or data fields.
Z	Displays all intervals in the result, even those with no entries. This option allows you to create a scale with no gaps. Use this option only with the S option.
*	Flags invalid subtotals, averages, standard deviations, maximums, and minimums if invalid data were used in the calculations.

Reserved Words

STAT2$ contains the number of lines ignored in all results for invalid or out-of-range key conditions.

STAT3$ contains the number of lines ignored in all results for invalid numeric, date, or time values. The value is meaningful only if you specified the V option.

Numeric Operators

Entry Count (=)	Counts the number of entries for each unique key.
Subtotal (+)	Subtotals values for each unique key in the key field.
Percentage (%)	Provides the entry counts for unique keys expressed as a percentage of the total entries in the field.
Subtotal Percent (+%)	Expresses the subtotal for each unique key as a percentage of the grand total of the key field.
Average (/)	Computes the average by subtotaling values and dividing by the number of entries.

Minimum (<) or (<D*n*) or (<T*n*)

Computes minimum value for each unique key. For date or time entries use <D*n* or <T*n*; *n* is the date or time format. For example, <D1 indicates to find the earliest date for each key (in format number 1, YYMMDD).

Maximum (>) or (>D*n*) or (>T*n*)

Computes maximum value for each unique key. For date or time entries use >D*n* or >T*n*; *n* is the date or time format. For example, >T0 indicates to find the latest time for each unique key (in format 0, HH:MM:SS).

Standard Deviation (!-) or (!+)

Computes standard deviation for a numeric field. Following are the operators and formulas used in calculating standard deviation. (*n* = entry count, and *x* = numeric field)

To Calculate Standard Deviation for a Sample:
Operator: !-
Formula: SQRT((SUM(*x***2)/(*n*-1))-(SUM(*x*)*SUM(*x*)/(*n**(*n*-1))))

To Calculate Standard Deviation for a Population:
Operator: !+
Formula: SQRT((SUM(*x***2)/(*n*))-(AVG(*x*)*AVG(*x*)))

Variance (!!-) or (!!+)

Computes the variance for a numeric field. Following are the operators and formulas used in calculating variance. (*n* = entry count, and *x* = numeric field)

To Calculate Variance for a Sample:
Operator: !!-
Formula: ((SUM(*x***2)/(*n*-1))-(SUM(*x*)*SUM(*x*)/(*n**(*n*-1)))

To Calculate Variance for a Population:
Operator: !!+
Formula: ((SUM(*x***2)/(*n*))-(AVG(*x*)*AVG(*x*))

Constant Label Operators

Constant labels (:R, :L, and :M) load fields with constants whose values are based on conditions that exist while the statement is executing.

Report Number (:R)	Places the input report number into the field.
Line Number (:L)	Loads the field with the input report line number.
Multiple Analysis Number (:M)	Loads the field with the corresponding analysis number. If you use more than one set of parameters (perform more than one analysis) in a statement, the statement creates a single result containing output from all of the analyses. This operator (:M) loads the specified field with the number of the analysis that is the source of the output line. For example, output lines created by the first set of parameters are loaded with a value of one.

Using the ICNT Run to Create a CNT Statement

You can use the Iterative Count (ICNT) run to create a CNT statement. For information on how to use the iterative runs, see the *Manual Functions Reference*.

CNT Statement Example

This statement finds the average of the Down field for each unique value in the Location field:

```
@CNT,0,a,42 '' 'Location','Down' |,1,'/' .
```

`0,a,42`	Searches report 42A0.
`''`	Uses no options.
`'Location'` `'Down'`	Identifies the fields to process.
`|`	Processes tab lines only.
`1,`	Identifies the Location field as the key field.
`'/'`	Identifies the Down field as the field to find the average for.

DAT (Date)

The Date (DAT) statement performs computations on dates (from 1944 to 2043) within a report or result and creates a result.

1100: On OS 1100 MAPPER Systems, you can process dates between 1964 and 2063.

Format

```
@DAT,c,d,r o cc ltyp,p .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report in which the computations are to be performed.
o	Options field (see Options).
cc	Column-character positions or names of the fields to perform computations on.
ltyp	Line type to process. If you specify the A option, you can leave this subfield blank but enter the comma.
p	Parameters:

Date Format

A	YMMDD	
B	YYMMDD	
C	DD MMM YY	
D	YDDD	
E	YYDDD	
F	DDMMYY	
G	MM/DD/YY	
I	MMDDYY	

Arithmetic

+	field to perform operations on
-	field to subtract
=	field in which to place result
K	constant field to add to or subtract from a field
:	field in which to place day of week
+n or *-n*	constant number to add to or subtract from a field

continued

continued

Field	Description
	The parameters subfield identifies which operations are performed in a DAT statement, how many date fields are used, their format, and which arithmetic functions are performed on each one. Always specify a result date field, such as B= or =. If you do not specify a format in either a + or - field, format B is assumed in both fields. If you specify a format for either a + or - field (but not both), that format is assumed for both fields. If you do not specify a format for the = field, the format specified for the + field (if one exists) is assumed; otherwise, the format specified for the - field is assumed.

Options

A Processes all line types.

T Converts a time field to decimal hours. The time field must be in the format HH:MM:SS or HHMM. Use the + parameter to specify the field to convert and an = parameter in the field in which to place the converted time. If you do not specify a result field, the converted time is placed into the original time field.

W Converts a date field into days of the week.

n Specifies the number of work days in a week for computations, where *n* is the number of work days.

Example 1: Subtracting Dates

Subtract the dates in the Produc Actual field from the dates in the Produc Plan field:

```
@dat,0,b,2 '' 'producactual','producplan',\
'shipdate' □,-,+,= .
```

or

```
@dat,0,b,2 '' 57-6,50-6,64-6 □,-,+,= .
```

where:

`''`	Uses no options
`'producactual'` `57-6`	Subtracts the date in the Produc Actual field (column 57 for 6 characters)

`'producplan'`
`50-6`
`+`
	from the Produc Plan field (column 50 for 6 characters)

`'shipdate'`
`64-6`
`=`
	and puts the difference in the Ship Date field (column 64 for 6 characters)

`□`	Processes tab lines only

Example 2: Converting Dates to a Different Format

Convert format B dates to format F dates:

```
@dat,0,b,2 '' 50-6,32-6 □,b,f= .
```

where:

`''`	Uses no options
`50-6` `b`	Converts the format b date in the Produc Plan field (column 50 for 6 characters)
`32-6` `f=`	to a format f date, and places it in the Produc Cost field (column 32 for 6 characters)
`□`	Processes tab lines only

Example 3: Subtracting Dates in Different Date Formats

Subtract format F dates in the Produc Cost field from the format B dates in the Produc Actual field:

```
@dat,-0 '' 32-6,57-6,77-3 □,f-,b+,= .
```

where:

`''`	Uses no options
`32-6` `f-`	Subtracts the format f date (from the previous example) in the Produc Cost field (column 32 for 6 characters)
`57-6` `b+`	from the format b date in the Produc Actual field (column 57 for 6 characters)
`77-3` `=`	and places the result in the Spc Cod field (column 77 for 3 characters)

Example 4: Adding a Constant Number of Days

Add three days to the dates in the Produc Actual field:

```
@dat,0,b,2 '' 57-6,64-6 □,b+3,= .
```

or

```
@dat,0,b,2 '' 'producactual','shipdate' □,b+3,= .
```

where:

`''`	Uses no options

`'producactual'`
`57-6`
`b+3`
	Adds three days to the format b date in the Produc Actual field (column 57 for 6 characters)

`'shipdate'`
`64-6`
`=`
	and places the resulting date in the Ship Date field (column 64 for 6 characters)

`□`	Processes tab lines only
`b+3,=`	Adds three days to the first field (column 57 for 6) and places the resulting date in the second field (column 64 for 6)

Example 5: Adding a Constant Number Contained in a Field

Add the constant in the Spc Cod field to the dates in the Ship Date field:

```
@dat,0,b,2 '' 77-3,64-6,'statusdate' □,k,+,= .
```

where:

`''`	Uses no options
`77-3` `k`	Adds the constant in the Spc Cod field (column 77 for 3 characters)
`64-6` `+`	to the date in the Ship Date field (column 64 for 6 characters)
`'statusdate'` `=`	and places the resulting date in the Status Date field (column 5 for 6 characters)
`□`	Processes tab lines only

Using the IDAT Run to Create a DAT Statement

This section on using the IDAT run applies only to the OS 1100 MAPPER System.

You can use the Iterative Date (IDAT) run to create a DAT statement. For information on how to use the iterative runs, see the *Manual Functions Reference*.

DC (Date Calculator)

The Date Calculator (DC) statement performs arithmetic calculations on dates (from 1944 to 2043) or times that reside in variables, or on literal dates or times.

Format

```
@DC eq vrslts .
```

Following is a description of the fields:

Field	Description
eq	Equations (separate with semicolons).
vrslts	Variables to capture the results of the equations in the order presented.

Comments

- Each equation in a DC statement creates a label (A through Z, consecutively) that you can use in subsequent equations in the same statement. You do not need to specify the format of a date represented by a label — it stays the same as the answer it represents and is remembered by the system.
- You cannot name your own labels, as you can with an Arithmetic (ART) statement. However, two constant labels are available for calculations based on the current date and time (TODAY and TIME).
- Your answer appears in the same format as the date you use in the equation unless you specify a format change.
- If the resulting value is not a valid date or time, the DC statement fills the variable with asterisks (*). It then treats the asterisks as zeroes in subsequent calculations.

Date and Time Formats

For each date and time you use in the equation, specify the format it is in first.

Format for Specifying Date and Time Formats

{D|T}*n*(*x*)

Following is a description of the subfields:

Field	Description
D	Denotes this is a date format.
T	Denotes this is a time format.
n	Number identifying the date format (0 through 8) or time format (0 through 3).
x	Variable containing the date or time, or the actual date or time.

Note: *If you add 1 to the time, it adds one to the hours.*

Use these date and time formats:

D0(*x*)	DATE0$	YMMDD
D1(*x*)	DATE1$	YYMMDD
D2(*x*)	DATE2$	DD MMM YY*
D3(*x*)	DATE3$	YDDD
D4(*x*)	DATE4$	YYDDD
D5(*x*)	DATE5$	DDMMYY
D6(*x*)	DATE6$	MM/DD/YY
D7(*x*)	DATE7$	MONTH DD, YYYY* *(constant label TODAY format)*
D8(*x*)	DATE8$	MMDDYY
T0(*x*)	TIME0$	HH:MM:SS *(constant label TIME format)*
T1(*x*)	TIME1$	HH:MM
T2(*x*)	TIME2$	HHMMSS
T3(*x*)	TIME3$	HHMM

* Since these formats contain spaces, enclose actual dates in apostrophes; however, do not enclose variables containing these formats in apostrophes.

Format for Changing Date and Time Formats

```
{D|T}{n|w}=expression
```

Following is a description of the subfields:

Field	Description
D	Denotes this is a date format.
T	Denotes this is a time format.
n	Number identifying the date or time format of the answer to ***expression***.
w	Specifies the answer of ***expression*** should be expressed as the day-of-the-week name.
expression	Calculations using literal data or variables (using the date and time formats listed earlier), constant labels, labels created previously in the same DC statement, or any combination of these.

Examples

V1 equals today's date plus 90 days in DATE7$ format:

```
@dc today+90 v1h18 .
```

V1 equals today's date in DATE1$ format; v2 equals current time in TIME0$ format:

```
@dc d1=today;time v1i6,v2h8 .
```

V1 equals the day of the week on which June 4, 1990 fell; v2 equals 900626 plus five days, changed to DATE7$ format; v3 equals the number of days difference between the date in v2 (label B) and today's date:

```
@dc dw=d7('june 4, 1990');d7=d1(900626)+5;\
b-today v1h10,v2h18,v3i4 .
```

V1 equals 900604 (v5) plus 60 days in DATE1$ format; v2 equals the day of the week that is the date in v1 (a); v3 equals the date in v1 (a) in DATE7$ format; v4 equals the date in v3 (c) minus two days in DATE7$ format:

```
@ldv v5i6=900604 .
@dc d1(v5)+60;dw=a;d7=a;c-2 v1i6,v2h10,v3h18,\
v4h18 .
```

DCR (Decode Report)

This section applies only to the OS 1100 MAPPER System, the U Series MAPPER System, and UNIX MAPPER Systems.

The Decode Report (DCR) statement returns an encoded report to a readable format. The decoded report becomes the current (-0) result.

Condition

You must use the same encryption key and encryption level used when the report was encoded.

1100: Encryption levels are not available on OS 1100 MAPPER Systems.

Format

```
@DCR,c,d,r,key[-level,,dspscram?] .
```

1100:

```
@DCR,c,d,r,key .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report to decode.
key	The one- to eight-character key that was used to encode the report.
-level	Encryption level. If the report was coded with a level, you must specify the same level here. **1100:** This subfield is not available on OS 1100 MAPPER Systems.
,,	Subfield not used at this time.
dspscram?	Is the scrambled data displayable? Y or N. Default = N. If you specified Y in this field when you encoded the report, you must specify Y here. **1100:** This subfield is unnecessary on OS 1100 MAPPER Systems.

Comments

- Use caution when encoding and decoding reports. Do not forget your key. A report cannot be decoded if you lose or forget the key because no record of it is kept by the MAPPER system. Your coordinator cannot tell you what it is.
- Because of high processing overhead, excessive use of this statement may affect system performance.
- System messages are produced in these cases:
  - You specified the incorrect key or encryption level.
  - The encoded report is corrupted.
  - The report is not encoded.

1100: On OS 1100 MAPPER Systems, the decoding process produces invalid characters under those circumstances.

Whenever the system produces these invalid characters, it displays a system message indicating an error and flags each invalid character in the result with an SOE character.

If the encoded report is corrupted, characters in the result shown as SOEs cannot be recovered.

- **1100:** If you are using a normal ASCII terminal, you may not be able to decode a report that was encoded from a National Character Set (NCS) terminal.

Examples

Decode report 10B0, which was encoded using the key jak:

```
@dcr,0,b,10,jak .
```

1100: The following example does not apply to OS 1100 MAPPER Systems.

Decode report 2C0, which was encoded using the key ins with level 6:

```
@dcr,0,c,2,ins-6 .
```

DEF (Define)

The Define (DEF) statement tests a variable, substring, or reserved word for its current contents or characteristics, or sets another variable to a value based on the option used.

Format

@DEF[,*o*,*lab*] *setv*,*testv* .

Following is a description of the subfields:

Field	Description
o	Option that specifies the information you want to determine about the variable. You can specify only one option. If you do not specify an option, the DEF statement returns a code that describes the contents of *testv*.
lab	Label to go to if *testv* is not defined. (To simplify your run logic, if you want the run to continue on the next line, use LIN1 in this subfield instead of a label number.)
setv	Variable in which the MAPPER system returns a value describing the variable or reserved words tested.
testv	Variable or reserved word to test. To test a member of an array, use *testv*[*n*], where *n* is the member (brackets are required).

Options

A Determines the alphabetic drawer. The variable to be tested must contain a valid numeric drawer (a positive octal number that identifies the drawer and cabinet within the system). The variable being set must be type A or H.

C Determines the number of significant characters (any characters other than spaces). Tab characters are significant characters.

I Determines the variable type. The system returns one of the following values to indicate the variable type:

Test Type	Setting
A	1
F	2
S	3
I	4
H	5
1100: O	6

K Determines whether the variable contains Kanji characters (*setv* contains 8 if so; otherwise, it contains 0 through 7 as described in the table at the end of this option list.

M Determines the cabinet. The variable to be tested must contain a valid numeric drawer.

N Determines the numeric drawer. The variable to be tested must contain a valid cabinet and drawer designation, for example 0B.

P Determines the packed size, that is, the size of the variable if it were loaded with a LDV,P statement.

S Determines the size.

T Determines the number of tab characters.

V Determines the name of the variable. For *testv*, enter an integer to represent the order in which the run created the variable. For example, the following statement loads <set> with the name of the variable that was fifth to be created in the run:

```
@def <set>s10,5
```

1100: On OS 1100 Systems, determines the name of the variable. If *testv* does not have a variable name assigned, the system returns spaces in *setv*. See Section 4 for a description of named variables.

If you do not use an option, the DEF statement sets the *setv* variable to a value from 0 to 7, depending on the contents of the *testv* variable:

setv	*testv*
0	All tab characters or spaces or both
1	All numeric characters
2	All alphabetic characters
3	Alphabetic and numeric characters
4	All special characters
5	Special and numeric characters
6	Special and alphabetic characters
7	Special, numeric, and alphabetic characters

Note that the contents of *testv* can include either just the characters indicated or both the characters and spaces.

Examples

Determine the kinds of characters in <chars> and return the code in <code>:

```
@def <code>i1 <chars> .
```

Determine the number of significant characters in <input> and return the value in <characters>:

```
@def,c <characters>,<input> .
```

Determine the size of <name> and return the value in <size>:

```
@def,s <size>i2 <name> .
```

DIR (Directory)

The Directory (DIR) statement loads variables with information about a data name from the system directory. With the DIR statement, you can determine whether a data name is valid and the kind of data the name represents (cabinet, drawer, or report).

Format

```
@DIR[,lab] name [vcabinet,vdrawer,vrpt,vhirptr] .
```

Following is a description of the field and subfields:

Field	Description
lab	Label to go to if the name is invalid.
name	Data name to define.
*vcabinet**	Variable to capture the cabinet number.
*vdrawer**	Variable to capture the drawer letter.
*vrpt**	Variable to capture the report number.
*vhirptr**	Variable to capture the higher report number if the data name defines a range of reports.

* A data name can represent a cabinet, a drawer, a report, or a range of reports.

Variables that do not apply for a particular data name are loaded with spaces unless they are defined as I type variables.

1100: Variables that do not apply for a particular data name are loaded with spaces.

Reserved Word

STAT1$ contains one of the following error codes if the data name supplied is invalid and the run continues at the label (the run errs if you do not use the label):

1 Data name was not found in the system directory.
2 Data name does not begin with an alphabetic character (A to Z).
3 Data name contains invalid characters (^, ;, /, comma, space, tab).
1100: Data name contains no alphanumeric characters (A to Z or 0 to 9).
4 Data name contains more than 16 characters.

Example

Load variables with information about data name order-status:

```
@dir,099 order-status <cabinet>a4,<drawer>h1,\
<report>a4,<hireport>a4 .
```

where:

099	Goes to label 99 if data name order-status does not exist
order-status	Specifies the data name to obtain information about
<cabinet>a4	Loads <cabinet> with the cabinet number of order-status
<drawer>h1	Loads <drawer> with the drawer letter
<report>a4	Loads <report> with the report number
<hireport>a4	Loads <hireport> with the higher range report number

DRW (Drawer)

The Drawer (DRW) statement loads variables with the following information: number of characters per line, next report to add, highest report number, and the report and line limits in a drawer.

Format

```
@DRW,c,d[,lab vcpl,vcs,vmfno,vmfn,vnxrd,vhirptd,vlnd,
vrptsd,vrlmt,vllmt] .
```

Following is a description of the subfields:

Field	Description
c,d	Cabinet and drawer to process.
lab	Label to go to if the drawer does not exist.
vcpl	Variable to capture the number of characters per line.
1100: *vcs*	Variable to capture the character set type: 0 LCS 1 FCS 2 FCSU
1100: *vmfno*	Variable to capture the MAPER file number (for example, 1, 2, or 3) where the drawer resides. (MAPER files are OS 1100 program files where the MAPPER database resides.)
1100: *vmfn*	Variable to capture the internal MAPER file name (for example, M00001, M00002, or M00003) where the drawer resides.
vnxrd	Variable to capture the next report number available in the drawer if a new report were added.
vhirptd	Variable to capture the highest report number used in the drawer.
vlnd	Variable to capture the total number of lines in the drawer.
vrptsd	Variable to capture the total number of reports in the drawer.
vrlmt	Variable to capture the report limit for the drawer.
vllmt	Variable to capture the line limit for the drawer.

Reserved Word

If a drawer exists, STAT1$ contains the number of the highest report created. If the drawer does not exist, STAT1$ contains 0, and the run goes to the specified label. If you did not use a label, the run continues at the next statement.

Example

Load v1 with the number of characters per line in cabinet 0, drawer B, and if that drawer does not exist, continue the run at label 5:

```
@drw,0,b,005 v1i3 .
```

DSM (Display Message)

The Display Message (DSM) statement displays a one-line message at the top of the screen. You can use the DSM statement in one of two ways:

- Display a message at the top of an existing screen.
- Display a report or result, placing a message at the top of the screen. You can display the report or result beginning at line one or at a specified line number.

Format

@DSM,*c*,*d*,*r*,*lmsg*[,*tabp*,*erase?*,*interim?*,*pdq*,*dc*,*dd*,*dr*,*dspl*,*dspf*] .

Following is a description of the subfields:

Field	Description
c,d,r	Report containing the message.
lmsg	Line number of the message in the report.
tabp	Tab character after which to place the cursor. Use a negative number to indicate the number of tab positions to move backwards. Maximum = 100. Default = the Roll field in the control line (position 0).
erase?	Erase the rest of the screen, Y or N. Default = N.
interim?	Continue the run without the user pressing **Resume**, Y or N. Default = N.
pdq	Number of lines to push down on the screen. Default = 0.
dc	Cabinet number of the report to display.
dd	Drawer of the report to display.
dr	Report number of the report to display.
dspl	Line number of the specified report at which to start the display. Default = 1.
dspf	Format of the report to display. (Omit this field if you are displaying a format defined with a FMT or SFC statement.)
	Format = 0 - 25. Default = 0, basic format.
	1100: Format = 0 - 6. Default = 0, basic format.

Comments

- The message to display resides in a report or result. You display it by specifying its line number.
- The DSM statement stalls the run until the run user presses **Resume**, unless you specify the interim display, in which case the run continues automatically. Note that the DSM statement clears the output area.
- The text displayed by the DSM statement appears as it does for MAPPER system messages.
- The screen display contains a function key bar. You can customize the function key bar so that you can include operations not available with the standard function key bar. See FKY in this section.

1100: On OS 1100 MAPPER Systems, the screen display may not contain a function key bar, depending on how your system or sign-on is set up.

- Less than (<) and greater than (>) signs are translated into blink characters.
- If you have a 132-character terminal, the messages are automatically centered.
- You can customize the report format by preceding the DSM statement with a Format (FMT) or Set Format Characters (SFC) statement. See FMT and SFC in this section.
- Do not place other run statements on the same line after the DSM statement. Other run statements following it on the same line are ignored. Put the next run statement on a new line.
- Capture input with INVAR$, INVR1$, INPUT$, or INSTR$.

Examples

Display a message (located on line 4 of the current result) at the top of the current screen:

```
@dsm,-0,4 .
```

Display report 2D0, place a message (located on line 4 of report 12A0) on the top of the screen, and position the cursor at the tenth tab position:

```
@dsm,0,a,12,4,10,,,,0,d,2 .
```

DSP (Display Report)

The Display Report (DSP) statement displays a report or result on the screen and clears the output area. If the display includes the standard function key bar, you can customize it to include operations not available on the standard function key bar. For information, refer to the Function Key (FKY) and Screen Control (SC) run statements.

If you want the run to finish automatically after displaying the report or result, use the Display Report and Exit (DSX) statement. With a DSX statement, the user exits the run when the display appears on the screen. The DSX statement uses the same fields and subfields as the DSP statement, except for the *interim?* subfield, which does not apply to the DSX statement. See the online help system (HELP,@DSX) for more information.

Format

@DSP,*c*,*d*,*r*[,*l*,*tabp*,*f*,*interim?*,*hold*,*msg*] .

Following is a description of the subfields:

Field	Description
c,d,r	Report to display.
l	Line number in the report at which to start the display.
tabp	Tab character after which to place the cursor. Use a negative number to indicate the number of tab positions to move backwards. Maximum = 100. Default = the Roll field in the control line (position 0).
f	Report format. (Omit this field if you are displaying a format defined with a FMT or SFC statement.) Format = 0 - 25. Default = 0, basic format. **1100:** Format = 0 - 6. Default = 0, basic format.
interim?	Continue the run, Y or N (Y causes the run to display the report and continue and N stalls the run until it is resumed). Default = N.

continued

continued

Field	Description
hold	Number of lines already on the display screen that you want to hold. Report is displayed beginning with the first nonheld screen line.
	1100: On OS 1100 MAPPER Systems, you can also enter the letter H to display the report headings.
msg	Message of up to 80 characters to display in the control line. Enclose the message in apostrophes. Default tab position = home position.
	1100: On OS 1100 MAPPER Systems, you can enter up to 79 characters to display on the control line.

Comments

- In debugging runs, the DSP statement displays the results of statements as they are processed to see if they produce the desired results. When they do, remove the DSP statements used for this purpose.
- In manually updating reports, the DSP statement displays reports to update during a run. If manual updating is to be an integral part of the run, leave in the DSP statements used for this purpose.
- The screen display contains a function key bar. You can customize the function key bar so that you can include operations not available with the standard function key bar. See FKY in this section.

1100: On OS 1100 MAPPER Systems, the screen display may not contain a function key bar, depending on how your system or sign-on is set up.

- You can customize the report format by preceding the DSP statement with a Format (FMT) or Set Format Characters (SFC) statement. See FMT and SFC in this section.
- Do not place other run statements on the same line after a DSP statement. Other run statements following it on the same line are ignored. Put the next run statement on a new line.
- The DSP statement clears the output area after executing.
- To continue the run after the display, press **Resume**.

- To continue the run after the display, press **Resume**.
- The I/O and LLP counts are reset following a noninterim DSP statement.

Examples

The first DSP statement displays the current result, the second displays renamed result -3, and the third displays report 2B0.

```
@dsp,-0 .
@dsp,-3 .
@dsp,0,b,2 .
```

Display report 1C0, starting at line 8 and continuing the run after the display:

```
@dsp,0,c,1,8,,,y .
```

Display format 5 of report 1D0, starting at line 3:

```
@dsp,0,d,1,3,,5,,,' This line displayed on control line. ' .
```

DVS (Define Variable Size)

The Define Variable Size (DVS) statement creates variables equal to the size of report fields. For example, use the DVS statement to capture input parameters to be processed against report fields, or to build screens whose input fields must be the same size as report fields.

1100: The DVS statement creates a -0 result and releases any previous -0 result.

Format

```
@DVS[,c,d,r,lab] field[,field,...,field] v[,v,...,v] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report where the fields reside. Default = -0.
lab	Label at which to continue execution if the specified fields are not found.
field	Fields whose size you want to define (can be field names or column-character positions). Separate fields with commas.
v	Variables to define. Specify the variable name or number and type, for example, v1h, v2s, and so on. Separate variables with commas.

Comments

- Generally, you use a DVS statement with named fields, because a name does not directly specify the field size. The run defines the size of the variable when it executes. Any input parameter or screen using the affected variable adjusts to changes in field sizes.
- In a DVS statement, you specify the variable and its type; you do not specify the variable size. The run assigns a size to the variable equal to its corresponding report field.

Examples

Initialize a variable to the size of the Cust Code field in report 2B0, for use as an input parameter:

```
@dvs,0,b,2 'custcode' v1h .
@chg input$ v1 .
```

Initialize variables to the size of the Order Number and Ord Qty fields from the current -0, and create a screen using their sizes:

```
@dvs 'ordernumber', 'ordqty' <num>h,<qty>i .
@brk .
                Enter Order Number: □<num>,
                Enter Quantity: □<qty>,
@brk out,-0,2,23,1,1,y,,,p .
@chg input$ <num>,<qty> .
```

ECR (Encode Report)

This section applies only to the OS 1100 MAPPER System, the U Series MAPPER System, and UNIX MAPPER Systems.

The Encode Report (ECR) statement transforms report data into unreadable code. The report data cannot be displayed until the correct key is specified with the Decode Report (DECODE) function or the DCR run statement. (See DCR in this section.)

1100: On OS 1100 MAPPER Systems, the encoded report becomes the current (-0) result. You can display the encoded report, but it is not readable.

The ECR statement is particularly useful for highly sensitive reports and messages. It is more secure than read/write passwords because no one, including the coordinator, can decode the report without knowing the key.

Notes:

1. *Ask the coordinator whether the ENCODE feature is installed on your system.*
2. *Because of high processing overhead, excessive use of the ECR statement may affect system performance.*

Format

```
@ECR,c,d,r,key[-level,hdgs?,,dspscram?] .
```

1100:

```
@ECR,c,d,r,key[,hdgs?] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report to encode.
key	The one- to eight-character key used to encode the report. Valid characters are A-Z and 0-9.
	1100: On OS 1100 MAPPER Systems, since the *-level* subfield is not available to scramble the data more, use more characters for the key to make it more difficult for an unauthorized user to guess the key.
-level	Level at which to encode the report. Valid levels are 1-8. The higher the level, the more the data is scrambled. Default = 1.
	1100: This subfield is not available on OS 1100 MAPPER Systems.
hdgs?	Encode report headings? Y or N. Default = N.
,,	Subfield not used at this time.
dspscram?	Allow the scrambled data to be displayed? Y or N. Default = N.
	1100: This subfield does not apply to OS 1100 MAPPER Systems.

Comments

- Use an encryption level to specify the amount of scrambling. The higher the level of encryption, the more the data is scrambled. Note that the higher levels of encryption consume more system resources to process the report.

1100: Encryption levels are not available on OS 1100 MAPPER Systems.

- Specify the *dspscram?* subfield to allow encoded reports to be transferred to the OS 1100 MAPPER System. If you enter a Y in this subfield, the system uses the encryption process used on the OS 1100 MAPPER System, allowing the report to be decoded using the OS 1100 MAPPER System.

1100: When encoding reports on OS 1100 MAPPER Systems, the *dspscram?* subfield does not apply; the subfield is for encoding reports on other systems so that they can be decoded on OS 1100 MAPPER Systems.

- Use caution when encoding reports. Do not forget your key. A report cannot be decoded if you forget the key because no record of it is kept by the MAPPER system. Your coordinator cannot tell you what it is.
- You cannot move encoded data between drawers with different report widths because you will not be able to decode it. If a drawer width is changed, all encoded reports in that drawer will be corrupted and they will not be able to be decoded.

- **1100:** Do not update an encoded report. Any change to the encoded report corrupts the report and you will not be able to decode it. Protect your report from updates by using an update password (see the Password [PSW] function in the online help system [HELP,PSW]).
- You may not be able to decode data with a normal ASCII terminal if the data was encoded with a National Character Set (NCS) terminal. Also, a report containing special NCS characters cannot be encoded with a normal ASCII terminal.
- Encoding a report approximately doubles its size.

Examples

This example encodes report 10B0, using the characters jak as the encryption key. The report headings are also encoded:

```
@ecr,0,b,10,jak,y .
```


1100: The following example does not apply to OS 1100 MAPPER Systems.

This example encodes report 2C0, using the characters salplan as the encryption key at encryption level 3. If you display the result, the encoded data appears as scrambled data:

```
@ecr,0,c,2,salplan-3,,,y .
```

EL- (Element Delete)

This section applies only to the OS 1100 MAPPER System.

The Element Delete (-EL) statement to delete a standard OS 1100 program file or symbolic element, or a data file.

Condition

The file must be a sector-formatted file with no read or write keys.

Format

```
@EL-[,lab] qual,fn[,cyc,elt,ver] .
```

Following is a description of the fields and subfields:

Field	Description
lab	Label to go to if the run encounters an error. See the STAT1$ reserved word status codes following this table.
qual	Qualifier.
fn	Name of the file to delete, or the file name containing the element to delete.
cyc	File cycle.
elt	Element name.
ver	Version.

Reserved Words

STAT1$ contains the following status codes if the statement is not able to delete the file:

0	Requested element not found in specified file.
1	File does not exist.
2	File already assigned to MAPPER system.
3	File already assigned exclusively to MAPPER system.
4	File already assigned to another user.
5	File already assigned exclusively to another user.
6	File rolled out.
7	Facilities currently unavailable.
8	Private file, under different project-id.
9	Read or write restrictions on the file.
10	File is not sector-formatted mass storage file.
11	File is not program file (if element is specified).
12	File is a MAPPER file.
13	System I/O error.
14	Facility warning or reject.
15	Insufficient or improperly formatted statement.

STAT2$ contains the system message number. User this number in the Load System Message (LSM) statement to retrieve the text of the message.

Example

Delete the file myqual*myfile:

```
@el- myqual,myfile .
```

ELT (Element)

This section applies only to the OS 1100 MAPPER System.

The Element (ELT) statement copies a MAPPER report or result to a standard OS 1100 program file or symbolic element, or to a data file.

If the file you are copying is not a currently assigned file, the ELT statement assigns it with a maximum granularity of 262,143 tracks.

Condition

The file must be a sector-formatted file with no read or write keys.

Format

```
@ELT,c,d,r[,lab] qual,fn[,cyc,elt,ver,mapperf?,hdgs?,
cs,newcyc?] .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report to copy.
lab	Label to go to if the run encounters an error.
qual	Qualifier.
fn	File name to transfer the report to.
cyc	File cycle.
elt	Element name.
ver	Version.
mapperf?	MAPPER format, Y or N. Default = N.
hdgs?	Include headings, Y or N. Default = N.

continued

continued

Field	Description
cs	Character set of the new report: L Fieldata F ASCII (default) U ASCII, uppercase
newcyc?	Create a new cycle, Y or N. Y creates a new cycle (+1) for the file and ignores the entry in the *cyc* subfield. Note that the run errs if you try to create a new cycle of a nonexistent file.

Reserved Words

STAT1$ contains error codes:

1 File (relative cycle requested) does not exist.
2 File is already assigned to the MAPPER system.
3 File is already assigned exclusively to the MAPPER system.
4 File is already assigned to another user.
5 File is already assigned exclusively to another user.
6 File is rolled out.
7 Facilities currently unavailable.
8 Private file, under different project-id.
9 Read or write restrictions are on the file.
10 File is not sector-formatted mass storage file.
11 File is not program file (if the element is specified).
12 File is a MAPPER file.
13 System I/O error.
14 Facility warning or reject.
15 Insufficient or improperly formatted statement.
16 File is not a data file (if the element is not specified).
17 Cycle attempted on nonexistent file.
18 Attempted to write past end of file.

STAT2$ contains a line number identifying the system message. Use a Load System Message (LSM) run statement to read the message. Place the number in STAT2$ in the *msgno* subfield in the LSM statement.

Example

This example creates myqual*myfile (qualifier*file) and copies the data from report 2B0 to the file. If there is an error and the system cannot copy the data, the run goes to label 99.

```
@elt,0,b,2,99 myqual,myfile .
```

FDR (Find and Read Line)

The Find and Read Line (FDR) statement finds a line containing data you specify. Follow the FDR statement with one or more Read Line Next (RLN) statements or a Read Line (RDL) statement, to read the line or lines.

Format

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

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report in which to find (and later read) a line.
l	Line number at which to start the scan.
q	Number of lines to scan. Default = all lines.
lab	Label to go to if no lines or data are found.
o	Options field (see Options).
cc	Column-character positions or names of the fields to scan.
ltyp	Line type to scan. If you specify the A option, you can leave this subfield blank, but enter the comma.
p	Find parameters.
vrpt	Variable to capture the report number where the first find was made.
vlno	Variable to capture the line number where the first find was made.

Options

A	Processes all line types.
C(S)	Distinguishes between uppercase and lowercase letters.
1100: C(*x*)	Alters the find process based on the character set order. Ordinarily the system processes the report based on the character set of the drawer. The C option allows you to choose the character set type on which to base the find. Use one of the following: C(F) Full character set (FCS) C(L) Limited character set (LCS) C(S) Strict comparison; distinguishes between uppercase and lowercase letters.
R*x*{-*y* |'*,y*}	Scans a range of reports from report *x* through report *y*; scan reports *x,y,...y*. (Note that the format R*x,y* requires apostrophes around the comma.)
	1100: On OS 1100 MAPPER Systems, you do not need to enclose the comma in apostrophes.
	Examples: r2','5 Scan reports 2 and 5 r2-10 Scan reports 2 through 10 r2-10','14 Scan reports 2 through 10 and 14
@	Searches for spaces.
/	Searches for the slant character.

Comments

- The FDR statement is different from a Find (FND) statement in that the FDR statement may be followed by an RLN statement instead of an RDL statement. It is generally more efficient to use an FDR/RLN combination instead of an FND/RLN combination.
- If a find is made, the report in which the find is made becomes the current -0. Any previous -0 result is released.
- See also RDC, RDL, and RLN in this section.

Example 1: Finding a Specific Line in a Drawer

Look for characters ip and read the order number on the found line:

```
@fdr,0,b '' 'stcd' □,ip <report>i6,<line>i6 .
@rln,<line>,099 'order' <ord>i .
```

where:

`''`	Uses no options
`'stcd'`	Looks in the St Cd field (column 2 for two characters)
`□`	Processes tab lines only
`ip`	Looks for the characters ip
`<report>i6`	Captures the report number where the find was made in <report>
`<line>i6`	Captures the line number where the find was made in <line>
`rln,<line>,099 'order' <ord>i`	Captures the order number on the found line (in <line>) in <ord>

Example 2: Finding a Line in a Report

Look for the word green and read the order number on the found line:

```
@fdr,0,b,2,6,100,099 '' 15-5 □,green ,v1i6 .
@rln,v1,099 39-5 v2i5 .
```

where:

6	Starts the scan at line 6
100	Scans 100 lines
099	Goes to label 99 if no finds are made
''	Uses no options
15-5	Scans column 15 for five characters
□	Processes tab lines only
green	Looks for the characters green
v1i6	Captures in v1 the line number where the find was made
rln,v1,099 39-5 v2i5	Captures in v2 the order number on the found line in v1

FIL (Create File)

The Create File (FIL) statement creates a file in either MAPPER format or native data file format using the data from a MAPPER report or result.

Format

```
@FIL,c,d,r[,mapperf?,hdgs?,lab] fn .
```

Following is a description of the field and subfields:

Field	Description
c,d,r	Report to copy.
mapperf?	Create the report in MAPPER format, Y or N. Default = N, creates the report in standard data file format. To create a file in binary format (from a binary report), enter a B.
	1100: Binary files are not available on OS 1100 MAPPER Systems.
hdgs?	Place report headings into the file, Y or N. Default = N.
lab	Label to go to if the report or drawer does not exist.
fn	Name of the file to create.
	1100: Include the full file name. The qualifier and file name must conform to OS 1100 file naming standards. Only alphabetic, numeric, and the special characters dollar sign ($) and minus (-) can be included. Refer to the *ECL Operations and Programming Reference* for complete information on files, elements, and cataloging. Default character set of the file to create = F (full character set; creates an ASCII file).
	U Series and UNIX: Include the full path name. The directory in which the file is to be created must exist. The maximum length of the file name is 14 characters; the maximum length of the path name and file name is 70 characters. File names can be uppercase or lowercase letters but are created (or must already exist) exactly as typed. For example, Abc must be accessed with an uppercase A and lowercase b and c.
	A Series: Include a usercode and pack family (optional). File names must be uppercase letters. If a usercode is not specified, the file is created under the user's usercode. If a pack family is not specified, the file is created on the pack family equated to DISK in the family statement. The file is created with security set to PUBLIC (I/O).

continued

continued

Field	Description
fn (cont.)	**BTOS:** Use the full path name: [*volume*]<*directory*>*filename*. The volume and directory in which the file is to be created must exist. BTOS does not default to the current directory when creating the file. File names are not case sensitive. **PC MAPPER:** Include the complete path name. The directory you specify for the file must exist. A file name cannot exceed eight characters in length; an extension cannot exceed three characters. Follow the DOS file naming conventions. If you use a reverse slant (\ ) to define a directory path in an FIL statement, it must be enclosed in apostrophes. This prevents the directory path from being misinterpreted as a MAPPER run statement that continues on the next line.

Comments

- If you create a file using the name of an existing file, the existing file is overwritten.
- If the file being loaded does not exist, the FIL statement creates the file.
- To keep information such as report number and date the report was last updated with the file, create the file with headings. This writes the date line into the file, so when you retrieve the file, the date line of the original report is part of the result.
- If the MAPPER report contains tab characters and you do not create the file in MAPPER format, the tab characters are translated to spaces. See Appendix C for information on translating tab characters to tab or other characters using the $TABA$ command.
- To control the format of the data in the files you create, use the data control commands described in Appendix C.
- **U Series and UNIX:** You can create the file in MAPPER format (which is unreadable by UNIX) or in standard UNIX file format (which can be processed by UNIX utilities).

- **A Series:** You can create the file in MAPPER format (which is unreadable by CANDE and MARC and contains special control characters) or in standard data file format (which can be processed by CANDE, MARC, and other A Series edit utilities).
- See the *A Series I/O Subsystem Programming Reference* for a description of valid file names.
- **BTOS:** If you use a variable for a BTOS file name, pack the variable to eliminate trailing spaces. A file name with trailing spaces can cause an error when the FIL statement is used. For information about packing a variable, refer to the Load Variable (LDV) statement and its P option.

Example 1: Translating Characters Using the $TRNA$ Command

Translate ampersands (&) to spaces (ASCII octal code 040) and dollar signs ($) to question marks (?):

```
$trna$ &,040 $,'?'
```

Example 2: 1100: Creating a Standard OS 1100 File

Create a standard OS 1100 data file called myqual*myfile using the data in report 2B0, including report headings:

```
@fil,0,b,2,n,y myqual*myfile .
```

Example 2: U Series and UNIX: Creating a Standard UNIX File

Create a standard UNIX file called production in the test directory using the data in report 2B0, including the report headings:

```
@fil,0,b,2,n,y /test/production .
```

Example 2: A Series: Creating a Standard Data File

Create a standard data file called PRODUCTION on the TEST pack family, using the data in report 2B0, including the report headings:

```
@fil,0,b,2,n,y '(TEST)PRODUCTION ON TEST' .
```

Example 2: PC MAPPER: Creating a Standard Data File

This example creates a file called prodstat.new in the \test directory using the data in report 2B0, including the report headings:

```
@fil,0,b,2,n,y '\test\prodstat.new' .
```

FKY (Function Key)

The Function Key (FKY) statement customizes the function key bar that is displayed at the bottom of the screen during a display by the Display (DSP), Display Message (DSM), or Output Mask (OUM) statements. You can also use the FKY statement to customize the control line.

1100: On OS 1100 MAPPER Systems you cannot use the FKY statement to customize the control line.

Condition

Use Screen Control commands, such as FKEY, in a separate report or in the output area of your run to define the function keys, and with the exception of the OS 1100 MAPPER System, the control line. See SC in this section.

Format

```
@FKY[,c,d,r]
```

Following is a description of the subfields:

Field	Description
c,d,r	Report containing the Screen Control commands that create a custom function key bar, control line, or both, for the following DSP, DSM, or OUM statement to display. **1100:** On OS 1100 MAPPER Systems, you cannot use the FKY statement to customize the control line.

Comments

- The DSP, DSM, and OUM statements continue to use the report you specified until one of the following conditions occur:
  - The run terminates.
  - The run links to another run via the Link to Another Run (LNK) statement.
  - You designate another Screen Control report with the FKY statement.
  - You use another FKY statement with no report specified.
- To return control to your run, use the KEY screen control command.

Example

This example sets up Screen Control commands in the output area, performs a Break (BRK) statement to make them accessible as -0, then uses FKY and DSP statements to display reports with and without a customized function key bar:

```
@brk .
fkey,1,'Resume',rsm
fkey,2,'Paint',pnt
fkey,4,'Return',KEY
fkey,10,'Quit',^
@brk fky,-0 .
@dsp,0,c,1 . (Report 1C0 appears with a custom function key bar.)
@if fkey$ = 4 gto 010 ; .
        .
        . (other processing)
        .
@fky .
@dsp,0,b,2 . (Report 2B0 appears with the default function key bar.)
```

FMT (Format)

The Format (FMT) statement creates a display format for a following output display, such as that created by a Display Report (DSP) or Output Mask (OUM) run statement. You can select which fields to display by specifying either field names or column characters.

1100: On OS 1100 MAPPER Systems, you can also select columns to print the next time you call the Auxiliary (AUX) or Print (PRT) run statements.

Format

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

Following is a description of the subfields:

Field	Description
c,d,r	Report from which to display fields. Default = -0.
field	Fields to display (can be field names or column-character positions).

Comments

- You can use field names or standard column-character syntax to define report fields:
  - If you name the fields, the display includes the columns of each field as well as the character immediately following the field.
  - If you use the column-character positions, the run displays only the columns specified.
- The system always includes column 1, which contains the line type designator, in the format.
- You can list fields in any order; however, fields are always displayed in the same order they appear in the report.

FMT (Format)

The Format (FMT) statement creates a display format for a following output display, such as that created by a Display Report (DSP) or Output Mask (OUM) run statement. You can select which fields to display by specifying either field names or column characters.

1100: On OS 1100 MAPPER Systems, you can also select columns to print the next time you call the Auxiliary (AUX) or Print (PRT) run statements.

Format

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

Following is a description of the subfields:

Field	Description
c,d,r	Report from which to display fields. Default = -0.
field	Fields to display (can be field names or column-character positions).

Comments

- You can use field names or standard column-character syntax to define report fields:
  - If you name the fields, the display includes the columns of each field as well as the character immediately following the field.
  - If you use the column-character positions, the run displays only the columns specified.
- The system always includes column 1, which contains the line type designator, in the format.
- You can list fields in any order; however, fields are always displayed in the same order they appear in the report.

Examples

Display the St Cd, Ship Date, and Cust Code fields of the current -0:

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

Display column 2 for 3 characters, column 45 for 5 characters and column 64 for 7 characters from report 2B0:

```
@fmt,0,b,2 2-3,45-5,64-7 .
@dsp,0,b,2 .
```

FND (Find)

The Find (FND) statement scans vertically through one or more reports or a result for one or more items.

Format

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

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report in which to find the data.
l	Line number at which to start the scan.
lab	Label to go to if no lines or data are found.
o	Options field (see Options).
cc	Column-character positions or names of the fields to scan.
ltyp	Line type to scan. If you specify the A option, you can leave this subfield blank, but enter the comma.
p	Find parameters.
vrpt	Variable to capture the report number where the first find is made.
vlno	Variable to capture the line number where the first find is made.

Options

A	Processes all line types.
C(S)	Distinguishes between uppercase and lowercase letters.
1100: C(*x*)	Alters the find process based on the character set order. Ordinarily the system processes the report based on the character set of the drawer. The C option allows you to choose the character set type on which to base the find. Use one of the following: C(F) Full character set (FCS) C(L) Limited character set (LCS) C(S) Strict comparison; distinguishes between uppercase and lowercase letters
R*x*{-*y*|','*y*}	Scans range of reports from report *x* through report *y*; scan reports *x*,*y*,...*y*. (Note that the format R*x*,*y* requires apostrophes around the comma.) 1100: On OS 1100 MAPPER Systems, you do not need to enclose the comma in apostrophes. Examples: r2','5 Scan reports 2 and 5 r2-10 Scan reports 2 through 10 r2-10','14 Scan reports 2 through 10 and 14
@	Searches for spaces.
/	Searches for the slant character.

Comments

- A find is different from a search. A search creates a result, but a FND statement loads a variable with the line number where the find is made.
- If the specified data is found, the statement:
  - Renames the report where the find is made to -0. Note that any previous -0 result is released.
  - Loads variables with information.
- To update a single line, use a FND statement followed by a Write Line (WRL) statement (see WRL in this section).

Example 1: Searching a Drawer for an Item

Search for the letters ip in the St Cd field and capture the report and line numbers where the find is made:

```
@fnd,0,b '' 'stcd' □,ip v1i6,v2i6 .
```

where:

`''`	Uses no options
`'stcd'`	Looks in the St Cd field (column 2 for two characters)
□	Processes tab lines only
`ip`	Looks for the characters ip
`v1i6`	Captures the report number where the find is made in v1
`v2i6`	Captures the line number where the find is made in v2

Example 2: Searching a Report for an Item

Search for the word green in column 15 for 5 characters and capture the line number where the find is made:

```
@fnd,0,b,2,6,099 '' 15-5 □,green ,<line>i6 .
```

where:

6	Starts the scan at line 6
099	Goes to label 99 if no finds are made
''	Uses no options
15-5	Scans column 15 for five characters
□	Processes tab lines only
green	Looks for the characters green
<line>i6	Captures the line number where the find is made in <line>

Example 3: Searching for Spaces

Search a field for spaces and capture the line number where the find is made:

```
@fnd,0,b,2,,099 @ 25-6 □,@@@@@@ ,v1i6 .
```

where:

099	Goes to label 99 if no finds are made
@	Uses the @ option to find spaces
25-6	Finds spaces in the Serial Number field (column 25 for six characters)
□	Processes tab lines only
@@@@@@	Looks for spaces
,v1i6	Captures the line number where the find is made in v1

Using the IFND Run to Create an FND Statement

This section on using the IFND run applies only to the OS 1100 MAPPER System.

You can use the Iterative Find (IFND) run to create a FND statement. For information on how to use the iterative runs, see the *Manual Functions Reference*.

GTO (Go To)

The Go To (GTO) statement allows you to transfer control (branch) to another point in the run.

Format

@GTO {*lab*|END[,*n*,*n*,*n*]|LIN [+]*n*|LIN -*n*|RPX *r*} .

Following is a description of the fields and subfields:

Field	Description
lab	Line label number (may be a variable).
END[,*n*,*n*,*n*]	Call to go to the end of the run (terminates the run and displays the output area). For runs started with a Link to Another Run (LNK) statement, *n*,*n*,*n* are up to three integer status codes that the linked run can pass back to the original run. The original run can access the codes using the reserved words STAT1$, STAT2$, and STAT3$. The codes are ignored if the run was not started via LNK (see LNK in this section).
LIN	Current line.
[+]*n*	Integer representing the number of lines following the current line. 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.
-*n*	Integer representing the number of lines preceding the current line. 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.
RPX	Call to go to another run control report in the same cabinet and drawer and execute the run statements.
r	Report number of a run control report in the same cabinet and drawer (cannot be a result).

Note: *+n, -n, and r can be variables. Do not, however, type a minus sign in front of the variable. The plus sign is optional with the line number, so if v1 contained 5, you could use @gto lin v1. Here is another example: @gto lin v2, where v2 contains the value -5, is the same as saying @gto lin -5.*

Comments

- The GTO RPX statement executes the run statements in the specified run control report with these considerations:
  - The run control report being entered must be in the same cabinet and drawer as the run that has the GTO RPX statement.
  - All security checks for the first run must be met by the run control report being entered.
  - All variables established in the run having the GTO RPX statement are valid in the run being entered. Label names used in the calling run can be used in the RPX run.
  - The run being entered by a GTO RPX statement need not be registered. However, since the RPX run control report resides in a drawer especially for MAPPER runs, inform your coordinator that you intend to use RPX in the run as part of the run plan.
- For information on computed IF/GTO statements, see IF in this section.

Examples

Go to a line with a label equaling the contents of v6:

```
@gto v6 .
```

Terminate the run and display the contents of the output area:

```
@gto end .
```

Execute the run statements in report 2 of the same cabinet and drawer beginning at line 3:

```
@gto rpx 2 .
```

Continue executing two lines beyond the current statement line:

```
@gto lin +2 .
```

HSH (Hash)

The Hash (HSH) statement assigns a hash number within a specified range to a given piece of input. As long as the specified range remains the same, the statement always produces the same numeric value for a given piece of input.

Format

```
@HSH v=vld,min-max .
```

Following is a description of the subfields:

Field	Description
v	Variable to receive the hash value.
vld	Data to which the system assigns a hash number. Specify the data using a variable, constant, reserved word, literal, or any combination.
min-max	Range within which the system assigns a hash number, where *min* is the lowest number in the range and *max* is the highest number in the range.

Comments

- The HSH statement performs a routine to assign a numeric value to the input. As long as the range remains the same, the routine produces the same numeric value for a given piece of input.
- A hash value can serve as a numeric identifier to index information. Using a hash value rather than an alphabetic or numeric scheme to index data ensures a more even distribution of data.

Examples

Index employee information among reports:

```
@hsh <empinfo>=user$,1-100 .
```

Assign a hash number within a range of 1 to 100 to the combined input of the user-id and local site identifier:

```
@hsh <hash>=user$|lrsd$,1-100 .
```

IDU (Index User)

The Index User (IDU) statement creates a result containing an index of a specified number of lines from selected reports in a drawer. You can choose one of these or all three: reports created or updated by a specific user, reports with specific last update start and end dates, and reports in a range.

Format

```
@IDU,c,d[,q,user,sdate,endate,srpt,endrpt vrpts,vlines,
vrptsd,vhirptd] .
```

Following is a description of the subfields:

Field	Description
c,d	Cabinet and drawer to index.
q	Number of lines to display from each report. Default = all heading lines up to eight lines.
user	User-id to index or "all" to index all users. Default = current user.
sdate	First update date to be included in the date range (format DDMMMYY). Default = all. This is the date of the most recent update to the report, or the creation date if there are no updates.
endate	Last update date to be included in the date range (format DDMMMYY). Default = current date. This is the date of the most recent update to the report, or the creation date if there are no updates.
srpt	Start report number for report range. Default = 1.
endrpt	End report number for report range. Default = highest report number.
vrpts	Variable to capture the number of reports found.
vlines	Variable to capture total number of lines in the reports found.
vrptsd	Variable to capture the total number of reports in the drawer.
vhirptd	Variable to capture the highest report number in the drawer.

Examples

Index cabinet 0, drawer A, and create a result containing the first three lines from each report updated by the user newuser:

```
@idu,0,a,3,newuser .
```

Index a range of reports under the user-id of newuser:

```
@idu,0,b,7,newuser,01jan90,30no'v'90,5,10 v1i3,v2i9,\
v3i4,v4i4 .
```

where:

`0,b`	Indexes cabinet 0, drawer B reports
`7`	Displays seven lines of data from each report indexed
`newuser`	Indexes user-id newuser
`01jan90`	Indexes reports starting on January 1, 1990
`30no'v'90`	Indexes reports ending on November 30, 1990
`5`	Starts index at report 5
`10`	Stops index at report 10
`v1i3`	Captures number of reports found in v1
`v2i9`	Captures number of lines found in v2
`v3i4`	Captures number of reports in drawer indexed in v3
`v4i4`	Captures highest report number created in v4

IF (If Conditional)

The If Conditional (IF) statement tests the relationship between two or more values and specifies the statements to execute when the test condition is true or false.

Formats

Basic Format

```
@IF[,C] val1 op val2 stmt1 . ; stmt2 .
```

Logical OR

```
@IF[,C] val1 op val2,val3[,val4,...,valn] stmt1 . ; stmt2 .
```

Logical AND

```
@IF[,C] val1 op val2 & op val3 [& op val4 ...
... & op valn] stmt1 . ; stmt2 .
```

Computational IF/GTO

```
@IF[,C] val1 op val2,(lab)[,val3,(lab),...,valn,
(lab)] . ; stmt1 .
```

Following is a description of the fields and subfields:

Field	Description	
C	C option to distinguish between uppercase and lowercase characters.	
val1	Data to compare to another value. You can specify the data as a variable, constant, literal, or reserved word.	
op	One of the following relational operators:	
	= or EQ	Equal
	GE	Greater than or equal
	> or GT	Greater than
	LE	Less than or equal
	< or LT	Less than
	NE	Not equal
	NOT = or NOT EQ	Not equal
	NOT < or NOT LT	Not less than
	NOT > or NOT GT	Not greater than

continued

continued

Field	Description
val2	Value to compare to *val1*.
stmt1 .	Run statement or statements to execute when the condition is true. Include the space-period-space after the statement. To omit the statement, include only the space-period-space.
;	Starts the "else" clause of the condition.
stmt2 .	Run statement or statements to execute when the condition is false. Include the space-period-space after the statement. To omit the statement and terminate the IF statement, include only the space-period-space.
,val3,valn	A comma (logical OR) and a third value, fourth value, and so on.
& op val3 & op valn	An ampersand (logical AND), another relational operator, and a third value, fourth value, and so on.
,(lab)	A comma and label number or location in the run to go to if the IF statement condition is met (TRUE). Valid entries for *lab* (computed GTO), in addition to a label number, include the following: END — End of run LIN *+n* — Current line plus *n* lines LIN *-n* — Current line minus *n* lines RPX *r* — Report number *r* (another run control report), where *r* is a report number in the same cabinet and drawer
stmt2	Statement indicating the action to take if the IF statement condition is not met (FALSE).

Comments

- Use the IF statement for conditional branching or the execution of another statement.
- If the condition is true, the statements are executed until a branch is encountered or the equation is terminated with a period. If the condition is false, the statements following the semicolon are executed.

- Always include the space-period-space-semicolon-space-period-space to terminate the IF statement, even when you omit *stmt1* or *stmt2*. For example, the following IF statement specifies the statement to execute when the condition is true; when the condition is false, the system continues execution with the next line in the run:

```
@if <number> > 0 ldv,p <packed>=<number> . ; .
```

- The IF statement treats spaces and the following characters as zeros in type A variables: . + -
- In type H variables, characters must be identical to satisfy a true condition.
- 1100: The IF statement considers the contents of a type O variable as a decimal value, not octal.
- When you compare substrings of up to 18 characters, the IF statement considers the characters in the substring as type A variables.

Examples

If USER$ equals newuser, go to label 2; or else go to label 1:

```
@if user$ = newuser gto 002 ; gto 001 .
```

If v1 equals v2, load v1 with 1 and continue on the next line; or else release the screen:

```
@if v1 = v2 ldv v1=1 . ; rel .
```

If <val1> and <val2> are not equal, go to label 3; or else release the screen:

```
@if <val1> ne <val2> gto 003 ; rel .
```

If v21 is greater than v20, go to label 3; or else execute the IF statement:

```
@if v21 > v20 gto 003 ; if v21 < v15 inc v21 . ; .
```

If <num> equals 2 OR 4, go to label 1; or else continue:

```
@if <num> eq 2,4 gto 001 ; .
```

If v1 is greater than 0 AND less than 100, go to label 3; or else continue:

```
@if v1 > 0 & < 100 gto 003 ; .
```

If v22 is greater than 10 AND less than 50, go to label in v99; or else continue:

```
@if v22 > 10 & < 50 gto v99 ; .
```

This example uses two IF statements. If v1 equals A and if v2 is less than B, go to label 1; or else continue:

```
@if v1 = a if v2 lt b gto 001 ; .
```

The following examples use a computed IF/GTO sequence.

If <total> equals 30, go to label 1; if <total> equals 40, go to label 2; if <total> equals 50 OR 60, go to label 3; or else continue:

```
@if <total> = 30,(001),40,(002),50,60,(003) ; .
```

If v2 equals 4, go to the second line following current line, OR if v2 equals 5, go to the end of the run; or else continue:

```
@if v2 = 4,(lin +2),5,(end) ; .
```

If v1 equals 2, execute the Load Variable (LDV) statement (load v3 with the value 4), go to label 1 and continue; if v1 does not equal 2, go to label 2:

```
@if v1 = 2 ldv v3=4 gto 001 ; gto 002 .
```

Execute the LDV statement (load v3 with the value 4) only if v1 equals 2; in either case (TRUE or FALSE), go to label 2 and continue:

```
@if v1 = 2 ldv v3=4 ; gto 002 .
```

Note that in the previous example, the run goes to label 2 even if the condition of the IF statement is not met, because the GTO statement is the next logical statement to execute.

This example uses an unknown trailing substring. (The 0-3 specifies the last three characters; the starting column position is unknown.) If the last three characters of v1 contain the letters mon, go to label 25:

```
@if v1(0-3) = mon,(025) ; .
```

This example uses a known trailing substring. (The 3-0 specifies the known starting position of 3 for the remainder of the field.) If the characters beginning with character 3 through the end of the field contain the letters fri, go to label 26:

```
@if v1(3-0) = fri,(026) ; .
```

If the value of <test1> is equal to <count>, the value of <test1> is increased by 1; otherwise, <test1> is increased by 2:

```
@if <test1> = <count> inc <test1> . ; inc,2 <test1> .
```

IND (Index)

The Index (IND) statement creates a result containing a listing of a specified number of lines from all reports in a drawer.

Format

```
@IND,c,d[,q,lab] .
```

Following is a description of the subfields:

Field	Description
c,d	Cabinet and drawer to index.
q	Number of lines to display from each report. Default = heading lines (up to eight).
lab	Label to go to if the drawer does not exist.

Reserved Words

STAT1$ contains the number of reports in the drawer.

STAT2$ contains the total number of lines in the drawer.

Comments

- The result contains a date line and lists the number of reports in the drawer, the total number of lines in the drawer, the number of lines contained in each report, and a specified number of lines from each report.
- If you are indexing reports that are fewer than 80 characters wide, the information containing the number of lines from each report is added as trailer lines.

Example

Index cabinet 0, drawer A, and create a result containing the first four lines from each report in the drawer (if no report exists, go to label 2):

```
@ind,0,a,4,002 .
```

ITV (Input Variable)

The Input Variable (ITV) statement accepts all input from an Out Variable (OUV) or interim Output (OUT) display by loading variables with the output.

Format

```
@ITV[,lab] v[,v...,v] .
```

Following is a description of the subfields:

Field	Description
lab	Label to go to if the user presses a function key.
v	Variables in which to capture input (up to 40 variables).

Comments

- The OUV/ITV combination does not alter or affect the current -0.
- The ITV statement following an OUV or interim OUT stalls the run until the user presses a function key or **Transmit**. While the run is stalled, the ITV statement controls the terminal and all input.
- Normally, if output is on display and you enter data on the control line and press **Transmit**, MAPPER software interprets the input. With the ITV statement, any data entered on the control line is ignored by MAPPER software and interpreted by the ITV statement. You can capture data entered on the control line if you place a tab character at home position.
- Data starts from a tab character; data following the first tab character is captured in the first variable, data following the second tab character is captured in the second variable, and so on.

- The ITV statement resets the I/O and LLP count.
- The length of an individual input field varies depending on the variable length or by what follows the tab character. The length of each data input is determined by the tab delimiter, the length of the variable, or the width of the run user's terminal.
- If the run user presses a function key while the ITV statement is active and the statement contains a label, the run continues at the label. FKEY$ contains a number indicating the function key pressed (contains 0 if the run user pressed **Transmit** with the cursor positioned below the control line).

Examples

This example captures input from an OUT statement:

```
@    brk .
Product:|
Description:|
@    brk out,-0,2,3,1,1,y,y .
@    itv,010 <product>s10,<desc>s10 .
     .
     . (other processing)
     .
@010: . respond to function key
@    if fkey$ = 1,(020),2(030) .
```

This example captures input from an OUV display:

```
@    ouv,1,15 'what is the password?'|
@    itv <password>s6 .
```

JUV (Justify Variable)

The Justify Variable (JUV) statement reformats the contents of numeric variables. Normally, you reformat the contents of a variable before displaying the value on the screen or writing it to a report.

1100: On OS 1100 Systems, you can reformat the contents of numeric variables up to 18 characters in length.

Once justified, a variable retains that justification until its contents are altered.

Format

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

Following is a description of the subfields:

Field	Description
o	Options field (see Options). Use only one option.
v	Variable to reformat. If the variable you specify does not contain numeric data, the system does not change the variable.

Options

C Inserts commas in the integer portion of the variable every third digit, eliminates leading zeros, right-justifies the contents, and blank fills all remaining characters to the left of the number.

Notes:

1. *You must remove commas inserted by the C option before executing a Change Variable (CHG) or If Conditional (IF) statement against the variable. Use any other option to remove commas.*

   1100: *On OS 1100 Systems, you do not need to remove commas before using CHG or IF statements.*

2. *If inserting commas would expand the variable beyond the defined size of the variable, the variable remains unchanged.*

D Deletes all commas and leading zeros, and right-justifies the contents within the variable.

L Eliminates leading and trailing zeros, left-justifies the contents, and blank fills all remaining characters to the right of the number.

R Eliminates leading and insignificant trailing zeros, right-justifies the contents, and blank fills all remaining characters to the left of the number.

X Eliminates leading zeros, left-justifies the contents, and zero fills all remaining characters to the right of the number.

Z Eliminates insignificant trailing zeros, right-justifies the contents, and zero fills all remaining characters to the left of the number.

Examples

In the following examples, Δ stands for a blank character position.

Initialize <string> to a type A, 12-character variable with a value of 6543.210:

```
@ldv <string>a12=6543.210 . <string> = 6543.210ΔΔΔΔ
```

Insert commas in <string>:

```
@juv,c <string> . <string> = ΔΔΔ6,543.210
```

Delete commas from <string>:

```
@juv,d <string> . <string> = ΔΔΔΔ6543.210
```

Left-justify the contents of <string>:

```
@juv,l <string> . <string> = 6543.21ΔΔΔΔΔ
```

Right-justify the contents of <string>:

```
@juv,r <string> . <string> = ΔΔΔΔΔ6543.21
```

Expand the contents of <string>:

```
@juv,x <string> . <string> = 6543.2100000
```

Right-justify the contents of <string> and add leading zeros:

```
@juv,z <string> . <string> = 000006543.21
```

KEY (Function Key Input)

The Function Key Input (KEY) statement lets you obtain a number via the FKEY$ reserved word that indicates the function key the run user pressed to move on from a noninterim Display Message (DSM), Display Report (DSP), Output (OUT), or Screen Control (SC) display. Use a KEY statement before DSM, DSP, OUT, or SC statements.

Note: A noninterim display is a screen displayed by a run that remains on display until the run user transmits or resumes.

Use the KEY statement only with output displays that do not use a function key bar (for example, the DSP statement displays the default MAPPER function key bar) since the functions on the function key bar override the effect of the KEY statement. See SC and FKY in this section for information about using a function key bar.

Format

```
@KEY .
```

Reserved Word

FKEY$ contains a number indicating the function key pressed (contains 0 if the run user pressed **Transmit** with the cursor positioned below the control line).

Example

Request function key input with the KEY statement and display report 2B0 using the OUT statement:

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

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

LCH (Locate and Change)

The Locate and Change (LCH) statement scans specific column-character positions of a report for a specified target string and changes it to a specified replacement string, creating a result.

Format

```
@LCH,c,d,r[,l,lab] o cc tgtstr/replstr [,vlines,vrpt] .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report in which to locate and change data.
l	Line number at which to start the scan. (The LCH statement processes data lines only; if you specify a heading line, the scan begins with the first data line.)
lab	Label to go to if no target string is located.
o	Options field. The A, F, and M options are always assumed (see Options).
cc	Column-character positions or names of the fields to be scanned.
tgtstr	Characters to locate.
replstr	Characters with which to replace the target string.
,	A skipped subfield. (This subfield is not used in LCH statements. Type the comma before entering the other variables.)
vlines	Variable to capture the number of lines located that contain the target string.
vrpt	Variable to capture the report number where the target string is located.

Options

A — Processes all line types. If used alone, this option instructs the system to ignore the first character of the target string, and to change the line type of all lines having finds to the first character of the replacement string. The A, F, and M options are always assumed.

C — Distinguishes between uppercase and lowercase characters. Without this option, case is ignored.

1100: On OS 1100 MAPPER Systems, this option applies only to full character set (FCS) reports.

F — Processes all line types and uses the first character of the *tgtstr* and *replstr* subfields as part of the target and replacement strings. The A, F, and M options are always assumed.

1100: On OS 1100 MAPPER Systems, the F option processes all line types and locates and changes the entire string. It does not locate strings that start in column one. The A, F, and M options are always assumed.

M — Treats the first character of the target string as the line type designator. This option takes the place of a line type subfield (normally shown as *ltyp*). You must use the M option to locate a string beginning in column 1. The A, F, and M options are always assumed.

O — Creates a result containing the items found.

OU — Creates an update result, after which you can do one of the following:

- Delete the found lines from the report with a Delete (DEL) statement.
- Delete the found lines from the report and place them in a result (-0) with an Extract (EXT) statement.
- Make changes to the update result and blend the updated lines from the result back into the report with an Update (UPD) statement.

S*x* Starts the scan at line *x*, where *x* is a positive number.

S*x-y* Starts the scan at line *x* and stop at line *y*.

S*x',n* Starts the scan at line *x* and scan for *n* lines. Be sure to enclose the comma within apostrophes (').

1100: On OS 1100 MAPPER Systems, you do not need to enclose the comma in apostrophes.

T*x* Sets *x* to a transparent character to match any character in that position. Do not, however, use the transparent character in the first character position of the target string. The default transparent character is a space.

1100: On OS 1100 MAPPER Systems, if the replacement string is shorter than the target string, the rightmost characters are filled with the specified transparent character.

Example

Change the word execution to processing:

```
@lch,0,a,1,40,099 afm 2-79 execution/processing ,\
<lines>i4 .
```

where:

40,099	Starts scan at line 40 and goes to label 99 if no target string is located
afm	Uses A, F, and M options
2-79	Starts scan in column 2 for 79 characters
execution/ processing	Specifies target and replacement strings: each time the word execution is encountered, replace it with the word processing
<lines>i4	Captures the number of lines located that contain the target string in <lines>

LCV (Locate and Change Variable)

The Locate and Change Variable (LCV) statement finds and optionally replaces data within a variable.

With the LCV statement, you can locate a string, change a string, count the number of occurrences, or compare two variables literally.

An LCV statement is to variables what a Locate (LOC) or Locate and Change (LCH) statement is to data in reports.

Format

```
@LCV[,lab] o v tgtstr[/replstr vpos,voccs] .
```

Following is a description of the fields and subfields:

Field	Description
lab	Label at which to continue execution when no finds are made or when *n* occurrences are not found (see the Bn option). To continue execution at the label when a find is made, use the N option. (To simplify your run logic, if you want the run to continue on the next line, use LIN1 in this subfield instead of a label number.)
o	Option that determines how to search the variable for data (see Options).
v	Variable to be scanned. You can specify any type of variable, and you can specify a substring or an array member.
tgtstr	The data you want to locate within the variable (called the target string). You can specify the data using a variable, literal, constant, or reserved word. Within the target string, the transparent character is a wildcard character; that is, the transparent character matches any character in that position. The default transparent character is the space. Therefore, any spaces in the data are treated as wildcard characters. You can use the Tx option to change the default transparent character when you need to locate spaces.

continued

continued

Field	Description
replstr	The data (from 0 to 256 characters) to replace the target string. If you do not use the B*n* option, the statement replaces all occurrences of the target string with the replacement string. To completely remove the target string, enter nothing for the replacement string (for example, target/).
vpos	Variable to capture the character position within the variable of the first occurrence of the target string. With the B*n* option, *vpos* captures the character position of the *n*th occurrence of the target string within the variable.
voccs	Variable to capture the number of occurrences of the target string within the variable. This is useful when doing a change or locate with the B option.

Options

B{*n* | *n-x*} Bail out option — with the B*n* and B*n-x* option, indicates which occurrences of the target string are to be processed.

- B*n* Locates only the *n*th occurrence of the target string (variable ***vpos*** captures the character position within the variable of the nth occurrence), or locates and changes *n* occurrences (***vpos*** contains the character position of the first changed occurrence).
- B*n-x* Locates and changes from *n* through *x* occurrences of the target string, where *n* is the first occurrence to process and *x* is the number of subsequent occurrences from *n* to process. Variable ***vpos*** captures the character position of the first changed occurrence.

C Distinguishes between uppercase and lowercase characters.

L*x* Locates a specific line type where *x* is the line type. The first character in the variable must match the line type designator the statement stops processing variable.

M — Uses the transparent character (see the Tx option) as a mask to inhibit replacements. Characters in the variable that correspond to the positions of the transparent characters in the target string are not replaced with those in the replacement string.

N — Goes to the specified label when a find is made rather than when no finds are made.

Tx — Specifies the character to use as the transparent character. The transparent character is a wildcard character used to match any character in the corresponding position. You can also use the transparent character as a mask to inhibit replacements (see the M option). Default = space character.

Example 1: Locating Second Occurrence of an Item

Contents of v1 for this example:

```
v1=doghorsecowpigbirdcatcatcatcatmouse
                  |                  |
                column 15          column 34
                  |  locate columns  |
```

Locate the second occurrence of the word cat in v1 starting at column 15 for 20 characters:

```
@lcv,001 b2 v1(15-20) cat v2i6 .
```

where:

001	Goes to label 1 if fewer than two occurrences of the word cat are found
b2	Bails out on the second occurrence of the word cat
v1(15-20)	Scans v1 starting in column 15 for 20 characters
cat	Locates target string cat
v2i6	Captures the character position where the second occurrence of the word cat begins

After the statement executes, v2 contains 22 (the column in v1 where the second occurrence of the word cat was found).

Example 2: Counting Occurrences of an Item

Contents of v1 for this example:

```
CATdogCATDOGCATDogCATDOGCATdog
```

Count the number of times the word dog occurs in v1:

```
@lcv b99 v1 dog ,v3i6 .
```

where:

b99	Bails out on the 99th occurrence of the word dog
v1	Scans v1 (the entire variable)
dog	Locates target string dog
v3i6	Captures the number of occurrences of the word dog in v3

In this example, the bail-out count was set to a value higher than the number of occurrences of the word dog because we did not want to bail out. V3 continues to count each occurrence of the word dog. After the statement executes, v3 contains 5.

Example 3: Changing Character Strings

Contents of v1 for this example:

```
*CATDOGCATDOGCATDOGCATDOGCATDOG
```

Change the second, third, and fourth occurrences of the word DOG to cat in v1:

```
@lcv l*b2-3 v1 DOG/cat v2i6 .
```

where:

`l*`	Makes the change only if the first character of v1 is an asterisk (the line type indicator)
`b2-3`	Changes target string starting at the second occurrence for three occurrences
`v1`	Scans v1
`DOG`	Locates target string DOG
`cat`	Changes target string DOG to cat
`v2i6`	Captures character position of first occurrence of target string changed

Since the first character of v1 matches the character specified in the L option, the change is made. V2 equals 11 and v1 contains:

```
*CATDOGCATcatCATcatCATcatCATDOG
      1     2     3     4     5
          v2=11
```

Example 4: Comparing Strings

Contents of <string1> for this example:

```
abc123
```

Contents of <string2>:

```
abc+++
```

Compare <string1> to <string2>:

```
@lcv,001 '' <string1> <string2> .
```

where:

001	Goes to label 1 if <string1> is not equal to <string2>
''	Uses no options
<string1> <string2>	Compares <string1> with <string2>

Since <string1> is not equal to <string2>, the run goes to label 1.

Remember, if you use the N option, the run does *not* go to the label unless the two variables are equal.

Example 5: Masking Transparent Characters in the Replacement String

Contents of v1 for this example:

```
blackbox1*blackcan1*blackbag1*blackcup1
```

Use the M option to locate each occurrence of the string black followed by any three characters, followed by the number 1. Each time the locate string is found, the characters black are changed to green, the next three characters remain unchanged, and the last character, 1, is changed to 2.

```
@lcv m v1 'black△△△1'/'green   2' v2i6,v3i6 .
```

where:

`m`	Specifies the mask option — transparent characters in the replacement string are not inserted into the variable being changed
`v1`	Scans v1
`'black△△△1'/`	Specifies the target string and change string delimiter (/)
`'green   2'`	Specifies the replacement string (characters with which to replace each occurrence of the target string)
`v2i6`	Captures the column of the first occurrence changed in v2i6
`v3i6`	Captures the number of occurrences changed in v3i6

V1 now contains:

```
greenbox2*greencan2*greenbag2*greencup2
```

Example 6: Using an Unknown Trailing Substring

Contents of v1 for this example:

```
feature010101
```

Use an unknown trailing substring and bail out after the second find:

```
@lcv,001 b2 v1(0-6) 01 v2i3 .
```

where:

001	Goes to label 1 if no finds are made
b2	Bails out after the second occurrence
v1(0-6)	Scans the last six characters of v1 (the 0-6 specifies the last six characters — the starting character position is unknown)
01	Locates the target string 01
v2i3	Captures the character position where the second occurrence of 01 begins in the specified substring (in this example, v2 contains 10 because the second 01 begins in the tenth column of v1)

Example 7: Using a Known Trailing Substring

Contents of <money> for this example:

```
$$$dollars
```

Use a known trailing substring to scan column 4 through the end of the field and changes the word dollars to the word yen in all occurrences:

```
@lcv,001 '' <money>(4-0) dollars/yen .
```

where:

001	Goes to label 1 if no finds are made
''	Uses no options
<money>(4-0)	Scans <money> beginning with column 4 through the remaining characters (the 4-0 specifies the known starting position of 4 through the end of the field)
dollars/yen	Changes the target string dollars to yen

LDA (Load Variable Array)

The Load Variable Array (LDA) statement defines a variable array and puts data into the array. Each variable in an array is referred to as a member of the array.

1100: The run control report must be in full character set (FCS or FCSU).

Format

```
@LDA[,o] nametypesize[n]=vld[,vld...,vld] .
```

Following is a description of the subfields:

Field	Description
o	Option that determines how data is put into the variables in the array (see Options).
nametypesize	Name of the variable array to initialize immediately followed by the variable type and number of characters allowed in each member of the array.
[*n*]	Number of members in the array. Brackets are required.
vld	Information to load in each member of the array (variables, literal data, constant, or reserved word).

Options

C — Centers data within the variable.

L — Left-justifies the data within each variable.

P — Packs data into each variable so that the variable contains only significant characters.

If you pack data that does not contain any significant characters, the variable into which you packed the data becomes a variable of 0 length. And, if you try to use a variable of 0 length in a run statement, the results are unpredictable.

To avoid creating a variable of 0 length, use the Define (DEF) run statement to determine the packed size of a variable before you pack it. See DEF in this section.

R — Right-justifies the data within each variable.

U — Converts all lowercase alphabetic characters to uppercase characters.

1100: W — Loads variables with the values of reserved words. (The reserved word values are left-justified. Make sure the variables are large enough to hold the values of the reserved words.)

Z — Zero fills each variable after the data is loaded. Right-justified variables are zero-filled to the left; left-justified variables are zero-filled to the right.

Comments

- Each variable in the array is counted as one variable of the total number of variables allowed on the system.
- 1100: Array members consume string variable space, regardless of the variable type.
- When you pass a variable array to a Call Subroutine (CALL) statement without specifying the array size ([*n*]), you can access all members of the array, but they count as only one of the 40 allowed to be passed.
- Use variable arrays to stack variables for the CALL statement.
- You cannot redefine the variable array, the size, or the type of any member of the array.
- Use brackets and numbers following the variable array name to access a member of the array. See the following example.

Example

Initialize an array of variables called <qty> as an integer array containing five array members, and load information into each member of the array:

```
@lda <qty>i6[5]=1,12,123,1234,12345 .
```

The members <qty>[1] through <qty>[5] contain the following information:

```
<qty>[1] =     1
<qty>[2] =    12
<qty>[3] =   123
<qty>[4] =  1234
<qty>[5] = 12345
```

LDV (Load Variable)

The Load Variable (LDV) statement initializes and loads variables. See also "Loading Variables Based on Content" in this subsection.

Format

```
@LDV[,o] v=vld[,v=vld,...v=vld] .
```

or

```
@LDV,o v[,v...,v] .
```

Following is a description of the subfields:

Field	Description
o	Option that determines how data is put into the variable or variables (see Options).
v	Variable into which you want to put data. To define or redefine the variable, include the variable type and size.
vld	Data you want to put into the variable. You can specify the data using a literal, constant, variable, reserved word, or any combination of these.

Options

C — Centers data within the variable.

1100: H — Tests the remote run link. Use the format `LDV,H v=rms`, where *v* is the variable to capture the result (0 = offline; 1 = online) and *rms* is the remote site number.

L — Left-justifies the data within the variable.

Packing Variables

With the P option, you can delete leading or trailing spaces from a variable. This is called packing a variable. Do not, however, pack a variable to contain no characters at all. If you do, other functions trying to use the variable may err.

Once you pack a variable to contain fewer than its original number of characters, you must reinitialize the variable to make it larger. If you try to place more than the original number of characters in a variable, you lose the extra characters.

Loading Variables Based on Content

The Q option allows you to load variables based on tab characters (or a specified delimiter) contained in the variable. The receiving variable is loaded with the data from the specified tab character (or delimiter) to the next tab set in the variable. Use the Q option when processing variables initialized using INVR1$ or INSTR$.

Format

@LDV,Q *rv=iv,n[(delim),rv=iv,n(delim)...,rv=iv,n(delim)]* .

Following is a description of the subfields:

Field	Description
rv	Receiving variable.
iv	Issuing variable.
n	Number of delimiters in *iv* to skip before loading data. Default = 1. This subfield must be a literal value.
(delim)	Character to use as a delimiter. Default = tab character. The parentheses are required.

LDV (Load Variable)

The Load Variable (LDV) statement initializes and loads variables. See also "Loading Variables Based on Content" in this subsection.

Format

```
@LDV[,o] v=v|d[,v=v|d,...v=v|d] .
```

or

```
@LDV,o v[,v...,v] .
```

Following is a description of the subfields:

Field	Description
o	Option that determines how data is put into the variable or variables (see Options).
v	Variable into which you want to put data. To define or redefine the variable, include the variable type and size.
v|d	Data you want to put into the variable. You can specify the data using a literal, constant, variable, reserved word, or any combination of these.

Options

C — Centers data within the variable.

1100: H — Tests the remote run link. Use the format `LDV,H v=rms`, where *v* is the variable to capture the result (0 = offline; 1 = online) and *rms* is the remote site number.

L — Left-justifies the data within the variable.

P — Packs the data in a variable so that the variable contains only significant characters. Use this option to remove unnecessary leading and trailing spaces.

If you pack a variable that does not contain any significant characters, the variable will contain no characters at all and other functions may not be able to use it. To avoid this condition, use the DEF run statement before packing a variable to determine its packed size. See "Packing Variables" in this section.

Q — Loads delimited data from a variable or reserved word into another variable. See "Loading Variables Based on Content" in this section.

R — Right-justifies the data within the variable.

S — Case sensitivity; differentiates between uppercase and lowercase character input when using the N option. Use the S option only with the N option; specify NS and not SN for this option.

U — Converts all lowercase alphabetic characters to uppercase characters.

1100: W — Loads the variable with the value of the reserved word. (The reserved word value is left-justified. Make sure the variable is large enough to hold the value of the reserved word.)

Z — Zero fills the variable after the data is loaded. Right-justified variables are zero-filled to the left; left-justified variables zero-filled to the right.

Comments

- Use an LDV statement to perform these tasks:
  - Initialize variables.
  - Load variables from other variables.
  - Load variables with literal data.
  - Load variables already initialized.
  - Load a substring of a variable already initialized. Use the substring format (***position-characters***) to load a substring.

- Load variables with the contents of reserved words.
- Load a variable with its own contents.

  When you load a variable with its own contents, you need only specify the receiving variable. This is particularly useful for packing variables. You can use, for example, **@ldv,p <var1>,<var2>** rather than **@ldv,p <var1>=<var1>,<var2>=<var2>**. Loading variables in this manner is also useful when centering, left justifying, right justifying variables, and initializing space filled variables without setting the characters equal to spaces.

- Whenever an LDV statement encounters a space or comma, it stops loading the variable with data. To place a space or comma in a variable without placing either character in a variable beforehand, enclose spaces and commas in apostrophes.
- You cannot do arithmetic computations with an LDV statement as you can with a Change Variable (CHG) statement (see CHG in this section). However, an LDV statement is faster and more efficient than a CHG statement for loading literal data and reserved words.

Loading Multiple Variables

In the following example, the first LDV statement loads v1 through v4; the second LDV statement then loads these four variables into v5:

```
@ldv <str1>a5=This,<str2>a3=is,<str3>a3=an,<str4>a8=example .
@ldv <string>s40=<str1><str2><str3><str4> .
```

<string> now contains the words "This is an example".

Packing Variables

With the P option, you can delete leading or trailing spaces from a variable. This is called packing a variable. Do not, however, pack a variable to contain no characters at all. If you do, other functions trying to use the variable may err.

Once you pack a variable to contain fewer than its original number of characters, you must reinitialize the variable to make it larger. If you try to place more than the original number of characters in a variable, you lose the extra characters.

Loading Variables Based on Content

The Q option allows you to load variables based on tab characters (or a specified delimiter) contained in the variable. The receiving variable is loaded with the data from the specified tab character (or delimiter) to the next tab set in the variable. Use the Q option when processing variables initialized using INVR1$ or INSTR$.

Format

@LDV,Q *rv=iv,n[(delim),rv=iv,n(delim)...,rv=iv,n(delim)]* .

Following is a description of the subfields:

Field	Description
rv	Receiving variable.
iv	Issuing variable.
n	Number of delimiters in *iv* to skip before loading data. Default = 1. This subfield must be a literal value.
(delim)	Character to use as a delimiter. Default = tab character. The parentheses are required.

LDV Examples with Various Options

The following statements are shown as if in sequential order in a run. The Δ stands for a blank character position.

Initialize v1 to 1:

```
@ldv v1i2=1 .
```

Initialize v1 to the letter a and v2 to 10:

```
@ldv v1a1=a,v2i2=10 .
```

Load v1 with This is dataΔΔΔΔΔΔΔΔ:

```
@ldv v1s20='This is data' .
```

Load v1 with ΔΔΔΔΔΔΔΔΔ2ΔΔΔΔΔΔΔΔΔΔ:

```
@ldv,c v1=2 .
```

Load v1 with 2:

```
@ldv,p v1=2 .
```

Load v1 with ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ1:

```
@ldv,r v1s20=1 .
```

Load v1 with 00000000000000000001:

```
@ldv,rz v1=1 .
```

Load v1 with ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ-1:

```
@ldv,r v1=-1 .
```

Load v1 with -00000000000000000001:

```
@ldv,rz v1=-1 .
```

Load v1 with ΔΔΔ20000000000000001:

```
@ldv,r v1(1-4)=2 .
```

Load v1 with 00020000000000000001:

```
@ldv,rz v1(1-4)=2 .
```

Load v1 with ABC△△△△△△△△△△△△△△△△:

```
@ldv,u v1=abc .
```

Load <cab> with the cabinet number, <drawer> with the numeric drawer number, and <report> with the report number of the last report or result processed or on display:

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

If NEWUSER is on station 12, v4 contains abcjdo12xyz:

```
@ldv,pw v4s20='abc'user$(1-3)stnum$xyz .
```

Use an unknown trailing substring to load v2 with the minutes and seconds (MM:SS) substring from v1 (the 0-5 specifies the last five characters, but the starting character position is unknown):

```
@ldv,w v1a8=time$ . v1 now contains hh:mm:ss
@ldv v2i5=v1(0-5) . v2 now contains mm:ss
```

Use a known trailing substring to load v2 with the seconds (SS) substring from v1 (from the previous example — the 7-0 specifies the substring beginning with the seventh character through the end of v1):

```
@ldv v2i2=v1(7-0) . v2 now contains ss
```

LFC (Load Format Characters)

The Load Format Characters (LFC) statement captures the format of the current report or result (-0). Use it, for example, when the run user might start your run with a displayed report that has been shifted or reformatted using the Create Temporary Format (VIEW) function (see the online help system [HELP,VIEW]).

Format

```
@LFC v .
```

Following is a description of the field:

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

Comments

- The variable receives a string of Xs and blank characters. The Xs stand for character positions displayed, similar to report 0.
- An LFC statement works only in runs registered as format sensitive.
- Use an Set Format Characters (SFC) statement after an LFC statement to set format characters.

Example

Capture the format of the current result in variable v1:

```
@lfc v1s80 .
```

The variable receives a string of Xs and blank characters. The Xs stand for character positions displayed, similar to the format lines in report 0.

LFN (Load Field Name)

The Load Field Name (LFN) statement loads variables with the names of report fields that correspond to the column-character positions supplied.

Format

```
@LFN[,c,d,r,tics?,lab] cc v[,v,...,v] .
```

Following is a description of the field and subfields:

Field	Description
c,d,r	Report from which to load field names. Default = -0.
tics?	Enclose the field name in apostrophes, Y or N. Default = N.
lab	Label to go to if the field name cannot be loaded.
cc	Column-character positions of the fields.
v	Variables to load with field names.

Reserved Word

STAT1$ contains one of the following codes in case of an error:

1 Report heading is improperly formatted for field names.
2 Columns supplied do not fall within field boundaries.
3 Field name in the report heading has no significant characters.
4 Field name is not unique in the report heading.
5 Field name was truncated because of the variable size, and the name is not unique in the report heading.

STAT2$ contains the number of the field in error.

Comments

- See Section 3 for a general description of named fields.
- The LFN statement is especially useful for converting an existing run to one that uses named fields. It is also useful for translating column position data, such as that obtained from the Output Mask (OUM) statement (see OUM in this section), into field names.
- You can have field names enclosed in apostrophes; this makes it easier for you to create run statements.
- If a specified variable is not large enough to contain an entire field name, any trailing characters in the name are truncated.
- If the specified columns do not represent an entire field, the statement loads the name followed by a partial field description.
- If the run cannot load a field name, it continues at the label.
- If the statement has no label and a field name cannot be loaded, the run continues at the next statement, loading the variable with the actual column-characters (for example, 2-2).

Example

Load field names from report 2B0:

```
@lfn,0,b,2,y 2-2,45-3 v1h18,v2h18 .
```

where:

`0,b,2`	Loads field names from report 2B0
`y`	Encloses field names in apostrophes
`2-2,45-3`	Gets names from column 2 for two characters and column 45 for three characters
`v1h18`	Loads v1 with the name from positions 2-2 (v1 = 'StCd')
`v2h18`	Loads v2 with the name from positions 45-3 (v2 = 'CustCode(1-3)')

LNK (Link to Another Run)

The Link to Another Run (LNK) statement executes another run in the MAPPER system.

Format

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

Following is a description of the subfields:

Field	Description
run	Name of the other run to start.
vld	Up to 40 input parameters consisting of variables, literals, reserved words, constants, or any combination of these. When the run starts, it uses their values to initialize variables via INPUT$. Pack or right-justify all variables.

Reserved Word

LINK$ contains 0 if the run is not started by a LNK statement or nonzero if the run is started by a LNK statement.

Comments

- A LNK statement differs from a Run Start (RUN) statement, in that when it starts a run that executes a GTO END statement, it returns control to the calling run. See RUN and GTO in this section.
- The system can transfer up to 40 input parameters, as well as the current result (-0), to the called run for processing.
- You cannot link to a run that is registered for a greater number of variables, labels, or variable characters.

1100: On OS 1100 MAPPER Systems, registering runs for different numbers of variables, labels, or variable characters is not available.

- The called run can return up to three numeric status codes via the GTO END statement, which you can use to signify whatever you like. The current -0 result is also available to the calling run. To capture the numeric status codes once control returns to the calling run, examine the reserved words STAT1$, STAT2$, and STAT3$.

- If the called run terminates with a Release Display (REL) statement, because of an error, or if the run user executes a Display Graphics (DSG), Display Report (DSP), Output Mask (OUM), Output (OUT), or Screen Control (SC) statement, the calling run does not resume.
- A run started by a LNK statement cannot itself contain another LNK statement unless you use a Clear Link (CLK) statement first to clear the original link.
- You can link to several utility runs, including chart runs and the Application Power Tools (APT) INFO run. See the online help system (HELP,@LNK) for details.
- Do not place other run statements on the same line after the LNK statement. Other run statements following it on the same line are ignored. Put the next run statement on a new line.

Example

Execute a run named test and pass the variables <first> and <second> to the linked run; when the linked run completes execution and passes control back to the original run, test the contents of STAT1$:

```
@lnk test,<first>,<second> .
@if stat1$ = 5 gto 100 .
```

The run named test returns control to the original run and passes three status codes to the run:

```
@gto end,5,10,20 .
```

LOC (Locate)

The Locate (LOC) statement locates a character string within a report or result and, with the O or OU option, creates a result.

Format

```
@LOC,c,d,r[,l,lab] o cc tgtstr [vcol,vlno,vrpt] .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report in which to locate a string of characters.
l	Line number at which to begin the scan. (The LOC statement processes data lines only; if you specify a heading line, the scan begins with the first data line.)
lab	Label to go to if no target string is located.
o	Options field. The A, F, and M options are always assumed (see Options).
cc	Column-character positions or names of the fields to be scanned.
tgtstr	String of characters to be located.
vcol	Variable to capture the number of the column preceding the column where the find starts.
vlno	Variable to capture the line number where the target string is located.
vrpt	Variable to capture the report number where the target string is located.

Options

A Processes all line types. If used alone, the first character of the target string is ignored. For example, if the target string is |abc (| represents a tab character), the A option looks for the string abc on all line types. The A, F, and M options are always assumed.

C Distinguishes between uppercase and lowercase characters.

1100: On OS 1100 MAPPER Systems, this option applies only to full character set (FCS) reports.

F Processes all line types and includes in the target the first character of the *tgtstr* subfield. The A, F, and M options are always assumed.

M Uses the first character of the target string as a line type designator. This option takes the place of a line type subfield (normally shown as *ltyp*). For example, if the target string is |abc (| represents a tab character), the M option looks for the string abc on tab lines. Use with the F option to find the string on all line types, even when the string starts in column 1. The A, F, and M options are always assumed.

O Creates a result containing items that are found.

Note: With the O option, the variable vcol *contains 0, the variable* vlno *contains the total number of lines that contain the target string (not the line number where the target string was located), and the variable* vrpt *contains the report number where the target string was located.*

OU Creates an update result; then you can perform one of the following operations:

- Delete the found lines from the report with a Delete (DEL) statement.
- Delete the found lines from the report and place them in a result (-0) with an Extract (EXT) statement.
- Make changes to the update result and blend the updated lines from the result back into the report with an Update (UPD) statement.

You cannot execute the LOC statement with the OU option against a result.

S*x* Starts scan at line *x*, where *x* is a positive number.

S*x*-*y* Starts scan at line *x* and stop at line *y*.

S*x*',*n* Starts scan at line *x* and scan *n* lines. The comma must be enclosed in apostrophes (').

> **1100:** On OS 1100 MAPPER Systems, you do not need enclose the comma in apostrophes.

T*x* Sets *x* to a transparent character to match any character in that position. Do not use it in the first character position of the target string. If you do not specify the T option, all blank characters are transparent. For example, the character string "A D" with the T option locates all A-space-space-D strings; without the T option, "A D" locates all four-character strings with A in the first character position and D in the fourth character position, such as ABCD, A 2D, and A%CD.

Example

Locate the characters fed:

```
@loc,0,b,2,,099 afm 2-79 fed <col>i3,<line>i5 .
```

where:

099	Goes to label 99 if no target string is located
afm	Uses the A, F, and M options
2-79	Starts scan in column 2 for 79 characters
fed	Locates the target string fed
<col>	Captures a number one less than the column number where the target string was located in <col>
<line>	Captures the line number where the target string was located in <line>

LSM (Load System Message)

The Load System Message (LSM) statement loads a variable with the contents of a system message.

Format

```
@LSM,msgno[,lab] vmsg .
```

Following is a description of the field and subfields:

Field	Description
msgno	Message number. If you are using this statement in an error subroutine, capture the message number with the reserved word XERR$ (see RER in this section).
lab	Label to go to if the message number does not exist.
vmsg	Variable in which to place the message. Specify 80 characters for the variable size so that the entire text of the system message fits in the variable.

Example

Load v2 with the message in message number v1:

```
@ldv,w v1i4=xerr$ lsm,v1 v2s80 .
```

or

```
@lsm,xerr$ <error>s80 .
```

LZR (Line Zero)

The Line Zero (LZR) statement creates or loads variables with information from line 0 of the designated report or result. The LZR statement does not create a result. The previously existing -0 result remains after the LZR statement executes.

Format

```
@LZR,c,d,r[,lab vlines,vcpl,vhdgs,vcs,vupds,vdept,vuser,vrpw,
vwpw,vlgn] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report from which to obtain information.
lab	Label to go to if the report number or drawer does not exist. Use this subfield alone to find out whether the report or drawer exists. (To simplify your run logic, if you want the run to continue on the next line, use LIN1 in this subfield instead of a label number.)
vlines	Variable to capture the number of lines.
vcpl	Variable to capture the number of characters per line.
vhdgs	Variable to capture the number of heading lines. For a result, this always equals 1.
1100: *vcs*	Variable to capture the number of the character set type: 0 = LCS, 1 = FCS, and 2 = FCSU.
1100: *vupds*	Variable to capture the number of updates to the report since it was created.
vdept	Variable to capture the department number if the report has a department-private read password.
vuser	Variable to capture the user-id if the report has a user-private read password.

continued

continued

Field	Description
vrpw	Variable to capture the read password, or the word LOCKED if the report has a read password.
vwpw	Variable to capture the write password, or the word LOCKED if the report has a write password.
vlgn	Variable to capture the number of the language with which this drawer was created. **1100:** On OS 1100 MAPPER Systems, the *vlgn* subfield does not apply.

Reserved Words

STAT1$, STAT2$, and STAT3$ contain the following information under these circumstances:

If the report exists:

STAT1$	Date of the last update in DATE1$ format YYMMDD
STAT2$	Creation date of the report in DATE1$ format YYMMDD
STAT3$	Save flag date, if one exists. If the save flag contains an invalid date, STAT3$ = 0.

If the report has never been updated:

STAT1$	0

If the report does not exist:

STAT1$	Highest report number in the drawer
STAT2$	(Disregard)

If the drawer does not exist:

STAT1$	0
STAT2$	(Disregard)

Comments

- The *vrpw* and *vwpw* variables contain the word LOCKED if the report has a read or write password. The word LOCKED is also indicated if the user is not signed on in department 2 (the MAPPER system coordination department).

1100: On OS 1100 MAPPER Systems, the word LOCKED is also indicated if the user is not signed on with the key user sign-on accessible to coordinators, which is not necessarily in department 2.

- If the report or drawer does not exist, the run continues at the label or, if you did not specify a label, the run continues at the next statement.
- If you specify a renamed result that is not previously renamed with a Rename (RNM) statement, the run errs; it does not go to the specified label.

1100: Under the same circumstances on the OS 1100 MAPPER System, the run continues at the specified label.

Example

Capture the number of lines in report 2B0, in <lines> (if that report or drawer does not exist, the run goes to label 99):

```
@lzr,0,b,2,099 <lines>i5 .
```

MCH (Match)

The Match (MCH) statement matches fields in two reports, then moves data from the issuing report to the receiving report, creating a result.

To create an update result that allows you to blend the changed lines back into the original report or delete lines from the original report, use the Match Update (MAU) statement. The MAU statement uses the same fields and subfields as the MCH statement. See the online help system (HELP,@MAU) for more information.

Format

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

Following is a description of the fields and subfields:

Field	Description
ic,id,ir	Issuing report.
rc,rd,rr	Receiving report.
lab	Label to go to if the issuing or receiving report is not sorted (valid only with the P option).
o	Options field (see Options).
icc	Column-character positions or names of the fields in the issuing report.
iltyp	Line type in the issuing report to match against. (If you specify the A option, you can leave this subfield blank, but enter the comma.)
ip	Parameters in the issuing report.
rcc	Column-character positions or names of the fields in the receiving report.
rltyp	Line type in the receiving report to match against. (If you specify the A option, you can leave this subfield blank, but enter the comma.)
rp	Parameters in the receiving report.

Options

A Matches all line types.

B Blends issuing and receiving report lines in a result. Data must be presorted. Do not use this option with the M or N options. This option does not apply to the MAU statement.

C(S) Distinguishes between uppercase and lowercase letters.

1100: On OS 1100 MAPPER Systems, this option applies only to full character set (FCS) reports.

D Deletes match information lines from the result. The D option is always assumed.

E Does not move blank fields from the issuing report. For matched items appearing in multiple lines of the issuing report, if the last item of the group is blank, the function does not move the data.

F Does not fill move fields in a no-match condition. This option does not apply to the MAU statement.

M Includes in the result only lines containing fields that match between the compared reports. Do not use this option with the B option. The M option is always assumed with the MAU statement.

N Includes in the result only lines containing fields that do not match between the compared reports. Do not use this option with the B option.

P Specifies that issuing and receiving reports are presorted by fields to be matched. Do not use this option with the Q option.

1100: See also Appendix D for information about which character set to specify when presorting.

Q Specifies quick execution of match data by assuming the data is presorted. When you specify the Q option, you guarantee that the reports are sorted, so the system does not check for unsorted fields. Do not use this option with the P option.

1100: See also Appendix D for information about which character set to specify when presorting.

S Includes in the result only lines containing fields that match, appearing in the same order as the issuing report.

Reserved Words

STAT1$ contains the number of lines in the receiving report that are matched or not matched, depending on options selected.

STAT2$ contains the total number of lines in the receiving report against which a match was attempted.

Comments

- If the match fields are not presorted, the system performs an internal sort on both reports. After processing, the receiving report is restored to its original order. (The maximum number of characters sorted internally is 64.) Use the P or Q option whenever possible.
- The MAPPER system places an update lock on the reports so that other users cannot update the reports while your run is matching data in them.

Example 1: Finding Lines That Do Not Match

Compare the same field name in reports 2B0 and 2C0, and find lines that exist in the receiving report but not in the issuing report:

```
@mch,0,b,2,0,c,2 n 'product' □,1 'product' □,1 .
```

or

```
@mch,0,b,2,0,c,2 n 15-9 □,1 2-9 □,1 .
```

where:

`0,b,2` `0,c,2`	Compares lines in reports 2B0 and 2C0
`n`	Uses the N option to include only lines that do not match

`'product'`
`15-9`
`1`
	Matches the Product Type field (column 15 for nine characters) in report 2B0

`'product'`
`2-9`
`1`
	to the Product Type field (column 2 for nine characters) in report 2C0

`□`	Processes tab lines in both reports

Example 2: Moving Columns in Lines That Match

Compare fields in reports 1C0 and 2B0, and move columns in lines that match:

```
@mch,0,c,1,0,b,2 m 2-9,'spacereq(0-3)' □,1,a \
15-9,77-3 □,1,a .
```

where:

0,c,1 **0,b,2**	Compares lines in reports 1C0 and 2B0
m	Uses the M option to include only matched lines in the result
2-9 **1**	Matches column 2 for nine characters in the issuing report
15-9 **1**	to column 15 for nine characters in the receiving report.
'spacereq(0-3)' **a**	Moves the last three columns of the Space Req field from report 1C0
77-3 **a**	to the Spc Cod field (column 77 for three characters) in report 2B0
□	Processes tab lines in both reports

MSG (Message to Control)

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

The Message to Console (MSG) statement sends a message to the system console.

Format

```
@MSG vld [vrsp] .
```

Following is a description of the field:

Field	Description
vld	Variables, literal data, reserved words, or any combination of these (up to 48 characters), to make up the message. If you use spaces, enclose the message in apostrophes.
1100: *vrsp*	Variable to capture the operator response (up to 60 characters). If you use this field, the run waits for a response before continuing.

The system console displays the message preceded by the following signal:

```
MAPPER*nnn*
```

where *nnn* is the run user's station number.

1100: On OS 1100 MAPPER Systems, the system console displays the message preceded by the following identifier:

```
MAPPER*4101:  MESSAGE FROM user-id AT MAPPER STATION nnn:
```

where *user-id* and *nnn* are the run user's user-id and station number, respectively.

The message itself appears on the next line of the console preceded by the characters MAPPER*4101+ .

Examples

The following statement is an example of a message:

```
@msg 'Please reload rpt 2B0 from 891103 tape.' .
```

1100: The following example applies only to OS 1100 MAPPER Systems:

This statement asks the operator whether to continue, and loads <response> with the response:

```
@msg 'Should my run continue?' <response>h3 .
@if <response> = no gto end ; .
```

NET (Network Sign On)

This section does not apply to the BTOS II MAPPER System.

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

Use the Network Remote (NRM) statement following the NET statement to use distributed MAPPER applications.

With the NRM statement, since the terminal remains under local control, the screen and keyboard retain the look and feel of the local site regardless of the kind of system to which the user is connected. This allows you to design distributed applications that let users process data on different systems transparently.

Format

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

Following is a description of the subfields:

Field	Description
net-id	Network identifier of the host system.
site-id	Site identifier of the remote MAPPER system. Default = first site configured for the specified network identifier.
rmu	User-id registered on the remote MAPPER system. Default = user-id of the user executing the run.
rmd	User department number for the remote MAPPER sign-on. Default = department number of the user executing the run.
rmpw	User-id password for the remote MAPPER sign-on. Default = password of the user executing the run.
trnrpt	Report that contains a translation table used to translate characters between the local and remote MAPPER sites. The translation reports must be in drawer F of the user's language cabinet. Default = no translation used.
mtr	Monitor transferred data, Y or N. Default = N.
lab	Label to go to if the connection fails.

Comments

- **U Series, UNIX, and PC MAPPER:** The user must be registered to execute the INTER_RUN run on the remote MAPPER system. Contact the coordinator to verify access to the INTER_RUN run.
- **1100:** The user must be registered to execute the INTER-RUN run on OS 1100 MAPPER Systems.
- **A Series:** The user must have a usercode on the remote system and must be registered to execute the INTER_RUN run on the remote MAPPER system. Contact the coordinator to verify usercodes and access to the INTER_RUN run.
- You must specify either the network identifier or the site identifier. If you do not specify the site identifier, MAPPER software defaults to the first site configured for the specified network identifier.

  If you specify only the site identifier, MAPPER software reads the network configuration report for the correct network identifier. Contact your MAPPER system coordinator for the network and site identifiers configured on your system.

Example

Sign on to remote host SYS15, site identity z, with sign-on newuser,7,newuser and monitor the sign-on to the remote site:

```
@net,SYS15,z,newuser,7,newuser,,y .
```

NOF (Network Off)

This section does not apply to the BTOS II MAPPER System.

The Network Off (NOF) statement signs the local MAPPER system caller off the remote MAPPER system after executing other network statements, such as Network Read (NRD) and Network Write (NWR). The NOF statement closes the communications connection with the remote system.

Format

```
@NOF .
```

NRD (Network Read)

This section does not apply to the BTOS II MAPPER System.

The Network Read (NRD) statement enables the user to read and return data from a remote MAPPER system to the local MAPPER system. Before using the NRD statement, you must first sign on to the remote system by using the Network Sign On (NET) statement.

Format

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

Following is a description of the field and subfields:

Field	Description
ic,id,ir	Issuing report on the remote MAPPER system (if you specify a result, you must also specify the cabinet and drawer, for example, 0,b,-2).
rc,rd,rr	Receiving report on the local MAPPER system. If you leave the *rr* subfield blank or enter 0, the issuing report is returned as a result (-0).
lab	Label to go to if an error occurs on the remote system.
vmsg	Variable to contain an 80-character system message if control goes to the label.

Example

Read report 2B0, on the remote MAPPER system and place the data into report 5B0, on the local MAPPER system (go to label 99 in case of an error):

```
@    nrd,0,b,2,0,b,5,099 <error>s80 .
@    dsp,-0 .
  .
  .  Other processing
  .
@099:ouv,1,1 <error> wat 50000 .
```

NRM (Network Remote)

This section does not apply to the BTOS II MAPPER System.

The Network Remote (NRM) statement executes runs and manual functions that reside on a remote MAPPER system. Before using the NRM statement, you must first sign on to the remote system by using the Network Sign On (NET) statement.

Format

```
@NRM "cmd" .
```

Following is a description of the field:

Field	Description
"cmd"	Run name or manual function call to execute at the remote site, including input parameters to be sent to the run or manual function. Enclose the request in quotation marks (").

Comments

- With the NRM statement, since the terminal remains under local control, the screen and keyboard retain the look and feel of the local site regardless of the kind of system to which the user is connected. This allows you to design distributed applications that let users process data on different systems transparently.
- To return to the local site, the run user can type two release characters (^^) and press **Transmit**.

  If the NRM statement started a run, you can have the remote run execute a Network Return (NRT) statement to return to the local site.

- The run containing the NRM statement remains active while the statement executes at the remote site.
- To maintain the transparency of a run executing at a remote site, be sure to include error and abort routines in the remote run. See RAR and RER in this section for information on abort and error routines.

  The routine should return the run user to the local site via an NRT statement in case of an error.

Example

Sign on to remote host SYS15, MAPPER site k, then use the NRM statement to execute a run named abc:

```
@net,SYS15,k,newuser,7,newuser .
@nrm "abc" .
```

NRN (Network Run)

This section does not apply to the BTOS II MAPPER System.

The Network Run (NRN) statement allows a local MAPPER system user to pass run statements to a remote MAPPER system to be executed. Before using the NRN statement, you must first sign on to the remote system by using the Network Sign On (NET) statement.

Format

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

Following is a description of the fields and subfield:

Field	Description
lab	Label to go to if an error occurs on the remote system.
"run statements"	MAPPER run statements to be executed on the remote MAPPER system. Remember to include the at sign (@) as the first character. Enclose the run statements in quotation marks. You can pass up to 256 characters in the run statements.
vmsg	Variable to contain an 80-character system message if control goes to the label.

Comments

- To process the results generated from an NRN statement, rename the results.
- Using the NRN statement to execute statements that load variables may cause conflicts with the INTER_RUN run that processes NRN requests. Do not use variables v1 through v15 in the NRN statement because they are reserved for the INTER_RUN run.

1100: On OS 1100 MAPPER Systems, the name of the run is INTER-RUN.

- **1100:** Passing a run statement that terminates control of a run causes the INTER-RUN run to end. Examples of this category of run statements are RUN and GTO RPX.

Example

Perform a Totalize (TOT) statement in the remote MAPPER system and renames the result -2 for further processing:

```
@nrn,099 "@tot,0,c,1 '' 18-6,25-7,65-8 ,+,-,= \
rnm -2" <error>s80 .
```

You can then use a Network Read (NRD) statement to read the result into a report on the local MAPPER system.

See NWR in this section for more examples.

NRT (Network Return)

This section does not apply to the BTOS II MAPPER System.

The Network Return (NRT) statement returns control to the local site that executed the run with a Network Remote (NRM) statement. Before using the NRT statement, you must use the NRM statement to start a run or manual function at the remote site.

Use the NRT statement in an error routine of a remote run to maintain the transparency of executing a run at a remote site. The NRT statement returns the run user to the local site in case of an error.

Format

```
@NRT ["cmd"] .
```

Following is a description of the field:

Field	Description
"cmd"	Run name or manual function call to execute at the local site, including input parameters to be sent to the run or manual function. Enclose the request in quotation marks ("). If no *"cmd"* is specified, the run returns control to the local site and continues at the statement following the NRM statement.

Example

A run named abc has been executed from another site. The last line of run abc includes an NRT statement to return control to the local site. Once control is returned to the local site the run that executed abc via an NRM statement continues executing at the next line:

```
@nrt .
```

NWR (Network Write)

This section does not apply to the BTOS II MAPPER System.

The Network Write (NWR) statement enables a user to send a report or result from a local MAPPER system to a remote MAPPER system. Before using the NWR statement, you must first sign on to the remote system by using the Network Sign On (NET) statement.

Format

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

Following is a description of the field and subfields:

Field	Description
ic,id,ir	Issuing report on the local MAPPER system.
rc,rd,rr	Receiving report on the remote MAPPER system.
lab	Label to go to if an error occurs.
vmsg	Variable to contain an 80-character system message if control goes to the label.

Receiving Report Entries

Enter the following information in the receiving report (*rr*) subfield:

Entry	Description
0	Duplicates the report within the drawer. STAT2$ contains the report number of the new report.
Receiving report number (positive)	Replaces the issuing report into the receiving report.
-1 through -8	Creates a renamed result, which you can use with the Network Run (NRN) statement.

Examples

Write report 2B0, to report 3B0, on the remote MAPPER system (if an error occurs, go to label 99):

```
@nwr,0,b,2,0,b,3,099 <error>s80 .
```

Send report 1C0, to the remote MAPPER system and rename it -1 in cabinet 0, drawer C. Use the NRN statement to execute a Match (MCH) statement between the renamed result and report 1D0, and rename the result -2. Send the -2 result back to the local MAPPER system via the Network Read (NRD) statement and display the result:

```
@nwr,0,c,1,0,c,-1,099 v1s80 .
@nrn,099 "@mch,0,d,1,-1 '' 12-9,31-8 ,1,a \
2-9,33-8 ,1,a rnm -2" v2s80 .
@nrd,0,c,-2,0,c,,099 v3s80 .
@dsp,-0 .
```

OK (Acknowledge Message)

The Acknowledge Message (OK) statement acknowledges any outstanding message queued to your station.

Format

```
@OK[,lab,vrsp] .
```

Following is a description of the subfields:

Field	Description
lab	Label to go to if no message is queued to your station.
vrsp	Variable to send as a response message.

Examples

Acknowledge an outstanding message and continue the run at the next statement (if there are no queued messages, go to label 3):

```
@ok,003 .
```

1100: The following example does not apply to OS 1100 MAPPER Systems.

As an example of how you might use an OK statement in a run, suppose your terminal beeps, notifying you of a message. You start a run that saves the message in a report.

First the run executes a Wait (WAT) statement with the M option to suspend the run while the system passes information to it that a message is waiting. If you started this run without a message waiting, the run waits for 1000 milliseconds, expecting the message, but then continues at the next line; otherwise, the run continues at the label, where it acknowledges the message and saves it in a report. Then the run displays information about the saved message. Here is the run:

```
@65:wat,m,60 1000 .
@brk .
   No message was received.
@gto end .
@60:ok dup,-O ldv,pw v1i4=cab$,v2a1=adrw$,v3i4=rpt$ .
@brk .
   The message is in report v3v2v1.
@brk out,-O,2,5,1,1,y .
```

1100: The following example applies only to OS 1100 MAPPER Systems.

As an example of how you might use an OK statement in a run, suppose your terminal beeps and you press **Message** to acknowledge the message. You start a run that duplicates the message into a report and captures the report number, drawer, and cabinet number where the system has added the new report. The run then acknowledges the message (OK) and displays the information. Here is the run:

```
@dup,-O ldv,pw v1i4=cab$,v2a1=adrw$,v3i4=rpt$ .
@ok brk .
   The message is in report v3v2v1.
@brk out,-O,2,5,1,1,y .
```

OS2 (Operating System Interface)

This section applies only to the Personal Computer MAPPER System.

The Operating System Interface (OS2) statement interfaces with the operating system and allows the execution of native PC operating system commands.

Condition

This statement should be used only by experienced MS-OS/2™ users.

Format

```
@OS2[,c,d,,,lab] o '[cmd]' .
```

Following are descriptions of the fields and subfields:

Field	Description
c,d	Drawer into which the result is to be returned.
lab	Label to go to if the command fails.
o	Options field (see Options). Enter options in lowercase characters. If you do not specify options, you must designate the options field with two apostrophes (' ').
cmd	Command to be performed. Standard path searching is done to find the command. Enclose commands containing embedded spaces in apostrophes ('). If you do not specify a command, designate its position with two apostrophes (' ').

MS-OS/2 is a trademark of Microsoft Corporation.

Options

-b Executes the command in the background and is minimized as an icon on the screen.

-f Causes the output of the command to be returned as a result in drawer A. If you enter a file name immediately following the -f option, the statement places a copy of the output from the command into that file. (You cannot use this option with the -w option.)

-w When the command is finished executing, the MAPPER system pauses until you enter the exit command in the active window to return to the active screen. (You cannot use this option with the -f option.)

Comments

- If you use the -b option but do not specify a command, the operating system command interpreter (CMD.EXE) session is displayed as an icon. You must open the icon, execute any commands, and exit back to your MAPPER session. Your MAPPER session shows a WAIT message until you exit from the CMD.EXE session.
- If you use a reverse slant (\ ) to define a directory path in an OS2 statement, it must be enclosed in apostrophes. This prevents the directory path from being misinterpreted as a MAPPER run statement that continues on the next line.

Example 1: Accessing the Operating System

This statement accesses the native OS/2 operating system (you must enter the apostrophes to preserve the positions of options and commands):

```
@os2 '' '' .
```

Example 2: Listing the Contents of a Directory

This statement lists the contents of the \mapper directory on drive C: and places the output in a cabinet 0, drawer B result:

```
@os2,0,b -f 'dir c:\mapper' .
```

OUM (Output Mask)

The Output Mask (OUM) statement displays a blank function mask on the basis of the headings defined for the specified cabinet and drawer. In the mask, the run user can enter options and parameters, which your run can capture.

Format

@OUM,*ic*,*id*[,*ir*,*if*,*rc*,*rd*,*rr*,*rf*,*title*] .

Following is a description of the subfields:

Field	Description
ic,id,ir	Report that contains the headings to be used in the mask (single mask) or of the issuing report (double mask). Default = report 0.
if	Format in which to display the mask (single mask) or of the issuing report (double mask). Default = basic format.
rc,rd,rr	Receiving report (double mask). Default = report 0.
rf	Format of the receiving report mask (double mask). Default = basic format.
title	Title (up to 12 characters) to be displayed above the mask.

Reserved Word

INMSV$ captures user input from the function mask screen. Use these guidelines:

- Use INMSV$ only with an OUM statement.
- Place INMSV$ before the variable names in the Change Variable (CHG) statement.
- Place the CHG statement before the OUM statement.
- Use string (type S) variables. You can use alphanumeric (type A) or Hollerith (type H) if the data fits in them.

Comments

- Use the blank function mask to load run parameter data into a variable. For a single mask, an OUM statement returns the option line and up to four decoded parameter lines. For a double mask, it returns the option line and up to two decoded parameter lines.
- The first variable initialized by INMSV$ contains the entire option line. In a single mask, you can capture up to four parameter lines. In a double mask, you can capture the two parameter lines allowed (one each for the issuing and receiving masks). Each parameter line requires three variables.
  - The first variable contains the starting column numbers of the fields where the run user entered parameters. Each starting column designation in the variable is three characters long, right-justified, and filled with zeros. The three-character designations for all starting columns are packed to the left in the variable. For example, parameter entries starting in columns 2 and 45 in the mask yield a variable containing 002045.

    Whenever you use a format other than basic, the column numbers are those defined in report 0 for the format. For example, if format 1 displays a field that actually starts in column 126, the variable contains 126 for the starting column number, not the column number where the field was displayed on the screen.

    Use a 120-character variable to hold the maximum 40 fields possible. The system fills unused character positions with blanks.
  - The second variable contains the corresponding field sizes, three characters each, packed to the left in the variable.
  - The third variable contains the entire parameter line expanded according to the format of the mask.

The run stalls until the user presses **Transmit**, then it continues with the data in the specified variables.

Note: *To use field names in your run rather than column positions, use the first and second parameter line variables with the Load Field Name (LFN) statement.*

- The screen display contains a function key bar. You can customize the function key bar so that you can include operations not available with the standard function key bar. See FKY in this section.

1100: On OS 1100 MAPPER Systems, the function key bar may or may not be available at your site.

Example 1: Using a Single Mask

Display a single function mask:

```
@chg inmsv$ v1s80,v2s120,v3s120,v4s132,v5s120,\
v6s120,v7s132 .
@oum,0,d,1,1,,,,,'TEST' .
```

where:

`@chg inmsv$`	Loads INMSV$ with variables
`v1s80`	Loads v1 with the option line
`v2s120`	Loads v2 with the first parameter line field starting columns
`v3s120`	Loads v3 with the first parameter line field sizes
`v4s132`	Loads v4 with the first parameter line
`v5s120`	Loads v5 with the second parameter line field starting columns
`v6s120`	Loads v6 with the second parameter line field sizes
`v7s132`	Loads v7 with the second parameter line
`oum,0,d,1,1`	Displays a blank function mask derived from report 1D0, format 1, to receive user input
`'TEST'`	Displays the title TEST above the mask

Example 2: Extracting Parameters

Capture the parameters entered from a function mask displayed by an OUM statement:

```
1. @     ldv v10i3=1 .
2. @     def,c v11i3,v2 .
3. @001: ldv v12i3=v2(v10-3) .
4. @     ldv v13i3=v3(v10-3) .
5. @     ldv v14s18=v4(v12-v13) .
  .
  .  Save variables here so loop
  .  can capture more information.
  .
6. @     INC,3 V10 .
7. @     IF V10 < V11 GTO 001 .
```

Here is a description of each line:

1. Initialize the column-character position.
2. Determine the number of fields selected based on the number of characters in v2.
3. Obtain the starting column number.
4. Obtain the field size.
5. Obtain the field from the parameter line.
6. Increase the column-character position by 3.
7. Loop back if there are more columns.

You now have all the options, column-character positions, and parameter entries in variables. Use these variables in statements such as Sort (SOR), Search (SRH), and Totalize (TOT), depending on the kinds of parameters the run user enters in the mask. An SOR statement does not get far, for example, if your variables contain search parameters.

Example 3: Using a Double Mask

Display a double function mask:

```
@chg inmsv$ v1s80,v2s120,v3s120,v4s132,v5s120,\
v6s120,v7s132 .
@oum,0,b,,,0,c .
```

where:

v1s80	Loads v1 with the option line
v2s120	Loads v2 with the issuing mask parameter line field starting columns
v3s120	Loads v3 with the issuing mask parameter line field sizes
v4s132	Loads v4 with the issuing mask parameter line
v5s120	Loads v5 with the receiving mask parameter line field starting columns
v6s120	Loads v6 with the receiving mask parameter line field sizes
v7s132	Loads v7 with the receiving mask parameter line
oum,0,b,,,0,c	Displays a double function mask derived from reports 0B0 and 0C0, basic format (using default report 0 and basic format for both reports)

OUT (Output)

The Output (OUT) statement clears the output area, clears any defined function key bar, and displays lines from a report or result on the screen. You can use it to display an entire screen or overlay an existing screen.

You may also use the Screen Control (SC) run statement to display data on the screen. The SC statement provides many features not available with the OUT statement. See SC in this section.

Format

```
@OUT,c,d,r,l,q[,outl,tabp,erase?,Interim?,pdq,protect,fxmt?,
outsp?,blink?,sn,lab] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report from which to display data.
l	Line number in the report or result from which the output is to start.
q	Number of lines to display.
outl	Line on the screen where the output is to start. Default = top line of the screen.
tabp	Tab character after which to position the cursor. A positive number is the number of tab positions to advance (maximum is 100) and a negative number is the number of positions to move back (maximum is 100) from the home position. Default = home position.
erase?	Erase the screen from the *outl*, Y or N. Default = Y.
interim?	Interim display (the run continues without the user resuming), Y or N. Default = N.
pdq	Number of lines to push down on the screen (push down quantity).
protect	Protected format option (see "Options for Protect Subfield" in this subsection).

continued

continued

Field	Description
fxmt?	Force transmit (the run continues and you can load variables with ICVAR$, INPUT$, INSTR$, INVAR$, or INVR1$), Y or N. Default = N.
outsp	Determines how data is sent to the screen. A, B, or space. Default = space. A — Sends data, including all spaces. B — Sends data, but not spaces. (If data exists on a line, only the columns containing new data are overwritten. The old and new data is blended together on the same line.) space — Sends the data, including spaces, if fewer than six consecutive spaces appear between the characters. If six or more spaces appear between the characters, the data is sent, but not the spaces. This allows the screen to be painted faster.
blink?	Change the less than (<) sign to a left blink character and the greater than (>) sign to a right blink character before displaying data, Y or N. Default = N. Hardware capabilities of supported terminals may not allow blinking. **BTOS and PC MAPPER:** Blinking display is not supported. This subfield is not used.
1100: *sn*	Station number where output is to be displayed. If you omit this field or specify 0 (zero), output is displayed at the station executing the run.
1100: *lab*	Label to go to if the output to another station cannot be completed successfully. If you omit this field and the run cannot be successfully completed, it is terminated with an error. See "Reserved Word" for the STAT1$ error codes.

Options for Protect Subfield

E Erases only unprotected fields. Use this option to clear input fields in a display.

1100: F Allows full-screen input. Use only with IBM® Series 3270 terminals.

I Edits input characters in unprotected fields by character type. Use one of the following edit codes in the first character position of the unprotected field in the output area. The numeric codes allow users to see the data as they type it in; the letter codes make the entered data transparent (it does not appear on the screen):

0/A or Space	Any data, no editing
1/B	Alphabetic data, no editing
2/C	Numeric data, left-justified
3/D	No data
4/E	Any data, right-justified
5/F	Alphabetic data, right-justified
6/G	Numeric data, right-justified

Numeric data includes plus and minus signs and the period.

The presence of the I option assumes output protect features. See the P option.

P Protects fields (for use when displaying data). Enter one of the following codes in column 1 of the output lines or immediately after the comma that ends an unprotected field:

0 or Space	Normal intensity
1	No intensity

IBM is a registered trademark of International Business Machines Corporation.

2	Low intensity
3	Blinking characters, alternating low and normal intensities

If protected and unprotected fields appear on the same line, start the unprotected field with a tab character and end it with a comma. To use different intensities on the same line, place an unprotected field between the protected fields.

BTOS: Hardware display intensity is not supported. For four-to-one output, leave line 3 blank. For five-to-one output, leave line 4 blank. The P option for the *protect* subfield is not used.

4 Fields are defined on four lines for one output line (see "Four- and Five-to-One Output" in this section):

- Line 1 contains M control characters.
- Line 2 contains N control characters.
- Line 3 contains emphasis characters.
- Line 4 contains the data to display.

(See your display terminal documentation for more information on controlling emphasis.)

5 Fields are defined on five lines for one output line (see "Four- and Five-to-One Output" in this section):

- Line 1 contains M control characters.
- Line 2 contains N control characters.
- Line 3 is the color line, with one of the codes listed in Table 7-10.
- Line 4 contains emphasis characters.
- Line 5 is the data to display.

Reserved Word

1100: If you use the *lab* subfield and the OUT statement cannot be completed, the run goes to the label. Examine STAT1$ for the status code:

1 Station does not exist or it is a batch port, remote run, or background station.

2 Station is not available because it is not currently connected to the MAPPER system.

3 No one is signed on at the specified station and *interim?* was not specified.

4 User at the specified station did not respond to the message wait signal within one minute.

5 User answered the signal and received the screen, but did another operation rather than supply input.

Comments

- An OUT statement places lines from a report or result on the screen, then clears the output area.
- The OUT statement stalls the run until the user presses **Transmit**, unless you specify a Y in the *interim?* or *fxmit?* subfields.
- In an OUT statement, you can specify:
  - Where to display the data

    1100: (includes displaying data on a different terminal)
  - Where to position the cursor
  - Which fields to protect
- You cannot use an OUT statement in a run started in background (BR statement) or from a remote site (RRN statement).

1100: On OS 1100 MAPPER Systems, you can use an OUT statement in background runs if you send the output to another terminal using the *sn* subfield.

- When the output is displayed, you can load input variables using INPUT$, INSTR$, ICVAR$, INVAR$, and INVR1$. See "Capturing Input" in Section 4 for more details about these reserved words. To resume the run after the output is displayed, press **Transmit**. You can also press **Resume** to resume the run, but the data entered on the screen is not captured.
- Put reserved words before the variable name in the Change Variable (CHG) statement, and be sure the CHG statement precedes the OUT statement (INSTR$, INVAR$, INVR1$).
- Initialize variables with INPUT$ following the OUT statement.
- Do not put other run statements on the same line after an OUT statement. The logic scan of the line terminates after executing the OUT statement.
- The I/O and LLP counts are reset following a noninterim OUT statement.

Examples

Display five lines of the current result, starting at line 2, on the first line of the screen:

```
@out,-0,2,5,1 .
```

The following statement uses several options:

```
@out,-0,4,1,1,,n,y,2 .
```

where:

-0	Displays lines from the current (-0) result
4	Starts at report line 4
1	Displays one line
1	Starts the display on the first line of the screen
n	Does not erase the screen
y	Specifies interim display (the run continues even if the user does not respond)
2	Pushes currently displayed data down two lines before displaying this line

Displaying Information At Another Terminal

This section on displaying information at another terminal applies only to the OS 1100 MAPPER System.

Use the *sn* and *lab* subfields to display information and obtain input from terminals other than the one executing the run. You can use background, batch port, and remote runs to solicit and obtain input just like normal runs, but you must send the output to a real terminal.

You cannot obtain exclusive use of any station. When more than one run is sending output to the same station, the output may be intermixed and not displayed in the same order as they were sent.

If the User Is At the Terminal

Sending information to a terminal where a user is signed on activates a message wait signal on that terminal and stalls your run. If the user responds to the signal, the information is displayed on the user's screen.

If you specify N or blank in the *Interim?* subfield, your run stalls until the user at the terminal presses any key. If you specify Y in the *Interim?* subfield, your run continues automatically.

If the user at the other terminal does not respond to the signal within one minute, your run either continues at the specified label or terminates with an error (STAT1$ = 4). If the user does respond to the signal, but enters a new command or function rather than supplying input, your run continues at the specified label or terminates with an error (STAT1$ = 5).

If the User Is Not at the Terminal

If no user is signed on at the specified station, you must specify a Y in the *Interim?* subfield. In this case, the message wait signal is not activated, the output information is displayed on the screen, and your run continues.

This is particularly useful if you want to use a terminal as a system monitor, with a scheduled background run periodically updating the display. If you do not specify an interim display, your run continues at the specified label or terminates with an error (STAT1$ = 3).

Four- and Five-to-One Output

If your display terminal lets you control emphasis for special editing and presentation techniques, you can use either the four- or the five-to-one output. Four-to-one is for monochrome displays and five-to-one is for color.

MAPPER software ignores the color feature on a monochrome terminal.

Four-to-one output consists of three edit code lines and one data line for each output line. Five-to-one output contains four edit code lines and a data line for each output line.

Each line in a four-to-one or five-to-one output contains an edit code character. These characters are:

Four-to-One	Five-to-One
M character	M character
N character	N character
Emphasis character	Color character
	Emphasis character

The following tables describe all possible combinations of the M and N characters for the four- and five-to-one displays. The first table shows the characteristics for the expanded edit code characters. The second illustrates the characteristics of the conventional edit code characters.

You cannot combine an M code from the expanded edit code character set with an N code from the conventional edit code character set, or vice versa.

The characteristics for the expanded M include the following:

Y (Yes) Entry	N (No) Entry
On Video	Off Video
Normal Intensity	Low Intensity
Tab Stop	No Tab Stop
Emphasis Protected	Emphasis Not Protected

Table 7-8 lists characteristics for the expanded M.

Table 7-8. Characteristics for the Expanded M

M Code	Protected Emphasis	Tab Stop	Field Changed*	Low Intensity	Video On
@	N	Y	Y	N	Y
A	N	Y	Y	N	N
B	N	Y	Y	Y	Y
C	N	Y	Y	Y	N
D	N	Y	N	N	Y
E	N	Y	N	N	N
F	N	Y	N	Y	Y
G	N	Y	N	Y	N
H	N	N	Y	N	Y
I	N	N	Y	N	N
J	N	N	Y	Y	Y
K	N	N	Y	Y	N
L	N	N	N	N	Y
M	N	N	N	N	N
N	N	N	N	Y	Y
O	N	N	N	Y	N
`	Y	Y	Y	N	Y
a	Y	Y	Y	N	N
b	Y	Y	Y	Y	Y
c	Y	Y	Y	Y	N
d	Y	Y	N	N	Y
e	Y	Y	N	N	N
f	Y	Y	N	Y	Y
g	Y	Y	N	Y	N
h	Y	N	Y	N	Y
i	Y	N	Y	N	N
j	Y	N	Y	Y	Y
k	Y	N	Y	Y	N
l	Y	N	N	N	Y
m	Y	N	N	N	N
n	Y	N	N	Y	Y
o	Y	N	N	Y	N

Y = Yes, N = No.

1100: * FIELD CHANGED used only for IBM 3270 terminals.

The characteristics for the expanded N include the following:

The characteristics for the expanded N include the following:

Y (Yes) Entry	N (No) Entry
Not Alpha Only	Alpha Only
Not Numeric Only	Numeric Only
Normal Field	Right-Justified Field
No Blink	Blinking
Normal Video	Reverse Video

Table 7-9 lists characteristics for the expanded N.

Table 7-9. Characteristics for the Expanded N

N Code	Reverse Video	Blink	Right-Justify	Number Only	Alpha Only
@	N	N	N	N	N
A	N	N	N	N	Y
B	N	N	N	Y	N
C	N	N	P	P	P
D	N	N	Y	N	N
E	N	N	Y	N	Y
F	N	N	Y	Y	N
G	N	N	P	P	P
H	N	Y	N	N	N
I	N	Y	N	N	Y
J	N	Y	N	Y	N
K	N	Y	N	Y	Y
L	N	Y	Y	N	N
M	N	Y	Y	N	Y
N	N	Y	Y	Y	N
O	N	Y	P	P	P
P	Y	N	N	N	N
Q	Y	N	N	N	Y
R	Y	N	N	Y	N
S	Y	N	N	P	P
T	Y	N	Y	N	N
U	Y	N	Y	N	Y
V	Y	N	Y	Y	N
W	Y	N	P	P	P
X	Y	Y	N	N	N
Y	Y	Y	N	N	Y
Z	Y	Y	N	Y	N
[	Y	Y	P	P	P
\	Y	Y	Y	N	N
]	Y	Y	Y	N	Y
^	Y	Y	Y	Y	N
~	Y	Y	P	P	P

Y = Yes, N = No, P = Protected input field; no input allowed.

Some of the intensity settings may not be valid because of limitations in the terminal hardware.

U Series: The following is a description of the attributes of each terminal:

- SVT-1210 — Reverse video, no distinction between normal and low intensity
- SVT-1220 and UVT-1224 — Normal intensity definition appears as high intensity
Normal/high intensity
Reverse video
Blinking characters
- PC — Refer to your PCU emulation documentation for PC capabilities

Color Characters

The third line in a five-to-one output contains the color code. Find the appropriate code in Table 7-10. Color codes sent to a monochrome terminal are ignored by MAPPER software.

Note: *Note that if you select the same color for background color and character color, the result is "invisible" characters.*

BTOS: Seven colors, plus black, are supported in the text mode. The only available background color is black. If you select black as the character color, the selected background color is used instead against a black background. If you select the same color for characters and background, green characters on a black background are used instead.

Table 7-10. Five-to-One Color Codes

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

Emphasis Characters

Emphasis characters consist of the column separator, the underscore, and the strike-through.

Because of hardware limitations of the supported terminals, MAPPER software ignores all emphasis characters. The emphasis characters are described here because the code character line is required for four-to-one or five-to-one displays.

1100: On OS 1100 MAPPER Systems, emphasis characters are sent to the terminal, but hardware capabilities vary among terminal types. See your display terminal documentation for more information on controlling emphasis.

Table 7-11 lists the applicable codes for emphasis characters.

Table 7-11. Emphasis Characters

Code	Strike Through	Underscore	Column Separator
space	-	-	-
!	-	-	Yes
$	-	Yes	-
%	-	Yes	Yes
(	Yes	-	-
)	Yes	-	Yes
'	Yes	Yes	-
_	Yes	Yes	Yes

For example, if you want your input fields to be displayed with underscores, select a $ from the table above. If you wish to create a vertical line, select a series of ! characters.

Now that you have had a brief look at edit codes and emphasis characters, you are ready to see how MAPPER software defines screens that use them.

Four-to-One Screens

These screens make full use of edit codes and emphasis characters. They allow you to specify the following:

- An M code
- An N code
- An emphasis character
- A displayable character

All can be specified for the same screen position.

You must place each of the codes and characters in the same column of four successive lines of source code. The M code must be on the first line, the N on the second, the emphasis character on the third, and the displayable character on the fourth.

Examples

For example, suppose you want the name of a field to be low intensity, reverse video, with protected fields. First, you would select an N from the M table, an S from the N table, and code the four lines as follows:

```
N                 (M byte)
S                 (N byte)
                  (Emphasis character)
State:__          (Displayable character)
```

If you would like the input field to contain a tab stop, have normal intensity, be left-justified, and allow alpha characters only, select a lowercase d from the M table and an uppercase A from the N table:

```
N        d        (M byte)
S        A        (N byte)
                  (Emphasis character)
State:__          (Displayable character)
```

When MAPPER software receives a four-to-one screen, it scans the source lines four at a time, vertically combines the four codes or characters of each column, and generates one line of output. The output line contains the edit codes, emphasis characters, and displayable characters ready for transmission to the terminal.

The following is an expanded edit code sequence:

```
D L L  (M byte)
A B C  (N byte)
```

Changing an unprotected field causes initialization of a new variable even though no tab character was specified.

Note: When redefining the editing code, even though no tab character was specified, the contents are placed into the next variable.

If you intend to display a four-to-one screen, enter a 4 as the protected format option on the OUT statement as follows:

```
@brk out,-0,2,16,1,1,y,,,4 .
```

Now you are ready to code a simple data entry screen. Suppose you need to enter the name, street, city, state, zip, and phone of a new customer. The state field should allow only alphabetic input. The input fields should be normal intensity, have tab stops, and be filled with underscores. The field labels should be low intensity and protected.

This example illustrates the four-to-one coding for a data entry screen.

```
1.   @brk .
2.   N       d                                      N
3.   S       @                                      S
4.           $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
5.   Name:
6.           d                                      N
7.           @                                      S
8.           $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
9.   Street:
10.          d                                      N
11.          @                                      S
12.          $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
13.  City:
14.          d N     d    N          d  Nd  Nd   N
15.          A S     B    S          B  SB  SB   S
16.          $$      $$$$$           $$$ $$$ $$$$
17.  State:       Zip:        Phone:(   )
18.  @brk out,-0,2,4,1,1,y,,,4 .
```

When the OUT statement executes, source lines 2 through 5 combine to form the first output line. Lines 6 through 9 combine to form the second output line, and so on. The following data entry screen is displayed:

```
Name:   _____________________________
Street: _____________________________
State:  __ Zip:_____  Phone (___)___-_____
City:   _____________________________
```

The underscored areas are unprotected and of normal intensity. Because the underscores are protected emphasis characters, they are continuous and are not overwritten by input from the keyboard. The field labels are protected and of low intensity. The State field allows only alphabetic input. The Zip and Phone fields allow only numeric input.

OUV (Out Variable)

The Out Variable (OUV) statement displays literal data and the contents of variables, constants, or reserved words on the screen.

Format

```
@OUV[,scl,col] vld .
```

Following is a description of the field and subfields:

Field	Description
scl	Line number of the screen at which to begin the display. Default = current screen line.
col	Column position at which to begin the display. Default = current column position. (Row 1 [*scl* = 1], column 1 [*col* = 1] is home position.)
vld	Data to display. The data can be a variable, constant, literal, reserved word, or any combination.

Comments

- Enclose literal data containing spaces in apostrophes.
- It is more efficient to use the OUV statement followed by a Wait (WAT) or Input Variable (ITV) statement than it is to use the Break-Output (BRK-OUT) sequence.
- Use the OUV statement while debugging your run to display the contents of variables while the run is executing.
- If an ITV statement immediately follows the OUV statement, the ITV statement stalls the run until input is entered.

Example

Display the contents of <output> on line 23 starting in column 10:

```
@ouv,23,10 <output>
```

PRT (Print)

The Print (PRT) statement prints reports and results on the system printer. The PRT statement does not create a result.

Format

```
@PRT,c,d[,r,dlnos?,f,prtsite,cys,all?,lsp,dstn,,hdgs?] .
```

1100:

```
@PRT,c,d[,r,dlnos?,f,prtsite,cys,all?,lsp,banner,formsid,hdgs?] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report to print.
dlnos?	Delete line numbers, Y or N. Default = Y.
f	Report format to be printed. Default = basic format. **1100:** On OS 1100 MAPPER Systems, default = basic format or, if a report was on display, the fields currently displayed by means of the VIEW function or those selected by an FMT or SFC run statement.
prtsite	MAPPER site at which to print the report (valid entries are A-Z). Default = local site. **1100:** On OS 1100 MAPPER Systems, device name or number of any printer available to the system. Default = first available printer in the printer group configured as the default.
cys	Number of copies to print (maximum of 64). Default = 1. **1100:** On OS 1100 MAPPER Systems, the maximum is 63 copies.
all?	Print all the reports of the specified drawer, Y or N. Default = N. Leave the report field blank if you use this subfield.
lsp	Line spacing, 1, 2, or 3. Default = 1.

continued

continued

Field	Description
dstn	Destination printer model. The model defines how the report is to be printed. You can set up special forms with different models. See the MAPPER system coordinator for available models. You can select a specific printer in this subfield, or you can let the MAPPER system select the default printer. **1100:** This subfield is not available on OS 1100 MAPPER Systems. **BTOS:** Uses a complete, valid BTOS printer name (for example, [LPT]) or spool name for the *dstn* field. Consult the MAPPER system coordinator for configured devices and spoolers.
1100: *banner*	Print banner name. Embedded tab codes or spaces are not permitted when using this subfield.
1100: *formsid*	Indicates the predefined or special forms identification. See also the PRINTFORM run (enter **printform,help**).
hdgs?	Print report headings? Choose one of the following: Y Print headings on first page only (default). N Do not print headings. A Print headings on every page. **1100:** When printing on special forms paper, the banner page, report headers, and end-of-report line are not printed. To override, enter *A.

Example

Print report 2B0:

```
@prt,0,b,2,,,,2 .
```

where:

0,b,2	Prints report 2B0
,,,,	Prints all columns without line numbers in basic format
2	Prints two copies

RAR (Register Abort Routine)

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

Since all variables, temporary results, and the current output area remain unaltered and are accessible to the abort routine, the routine can capture information currently being processed even though the user aborted the run.

Format

```
@RAR{,c,d,r lab | lab} .
```

Following is a description of the field and subfields:

Field	Description
c,d,r	Report containing an external routine. **1100:** The report containing the external routine must be in the same character set type as the calling run control report.
lab	Label in the report where an abort routine begins. To begin the routine at the first line of the external run control report, use LIN1 in this field.

Reserved Words

The following reserved words contain zero until a run aborts. The system resets them to zero whenever the run executes a noninterim display with a Display Graphics (DSG), Display Message (DSM), Display Report (DSP), Output (OUT), or Screen Control (SC) statement.

AXDRW$ contains the drawer letter of the run control report where the run aborted.

XDRW$ contains the drawer of the run control report where the run aborted.

XFUN$ contains the name of the last statement (the statement call) executed before the run aborted.

XLINE$ contains the line number in the run control report where the run aborted.

XRPT$ contains the report number of the run control report where the run aborted.

Comments

- If a run user aborts a run and the run has an abort routine registered, the system runs the abort routine and cancels any previously registered abort and error routines and update locks.
- Place the RAR statement as close as possible to the beginning of the run. If the user aborts the run before the run executes an RAR statement, the run has no abort routine to go to.
- The system cancels the abort routine when the run terminates, but you can also use the Clear Abort Routine (CAR) statement to cancel the routine.
- The abort routine cannot contain the statements GTO RPX, RAR, RER, or RUN. You can use an RSR statement only if it starts an internal subroutine.
- If a user presses **Abort** while an abort routine is executing, the run aborts.
- I/O and LLP limits and cabinet restrictions from the aborted run also apply to the abort routine.
- A noninterim display specified by any of the following statements in your abort routine continues normal run processing when a user tries to abort a run:

  DSG, DSM, DSP, OUT, SC

  This feature prevents users from terminating a run prematurely.
- The values of reserved words IO$ and LLP$ reflect the number of I/Os and LLPs used in the abort routine and the run.

1100: On OS 1100 MAPPER Systems, IO$ and LLP$ reserved words in an abort routine reflect the I/Os and LLPs in the abort routine itself.

- Abort routines registered in called routines (via the CALL statement) override previously registered abort routines; however, when control returns to the calling run, the system automatically reregisters the original abort routine.

1100: On OS 1100 MAPPER Systems, the system does not automatically reregister the original abort routine; you must reregister it yourself.

- If you do not register an abort routine in the called routine and a user aborts the run, control goes to the abort routine in the calling run.

1100: On OS 1100 MAPPER Systems, control does not go to any abort routines previously registered in the calling run.

Example

This statement registers the external abort routine at label 5 in report 4E0:

```
@rar,0,e,4 005 .
```

RDB (Run Debug)

This section does not apply to the OS 1100 MAPPER System. See "RDB (Run Debug): OS 1100" in this section for a description of the RDB statement as it applies to the OS 1100 MAPPER System.

The Run Debug (RDB) statement helps you locate and correct any errors in a run while it is executing. Use the RDB statement in a run instead of the manual function request to start the debugging at a specific point in a run. This is especially useful for debugging large or complex runs.

With the Run Debug (RDB) utility, you can step through your run to do the following:

- Display the contents of a variable or reserved word.
- Change the value of a variable.
- Display a line or range of lines in the run control report.
- Pause at a specified variable, run statement, or label.
- Make corrections to the run control report.

Condition

You must be the one who last updated the run control report; otherwise the system ignores the RDB request.

The run must not be registered in the Global Run Registration report if you want to execute RDB manually. (However, you can use the RDB statement in the run.)

Format

```
@RDB .
```

Manual Function Request

```
RDB run[,v,...v]
```

where *run* is the name of the run to debug and *v,...v* is data supplied to the run to be initialized via INPUT$.

RDB Commands and Function Keys

Use RDB commands and function keys **F1**, **F2**, and **F4** to view different kinds of information in your run. Table 7-12 describes the RDB commands and function keys.

Table 7-12. RDB Commands and Function Keys

Command/Function Key	Description
B[*Llab*|*@rfc*|*v*]	Stops the run at the point (breakpoint) specified in the parameter. You can have 13 breakpoints active at one time. If you do not specify any parameters, the B command stops on the lines examined using the E command (maximum of 10 lines set for breakpoint). *Llab* — Letter L followed by the label number after which to interrupt the run (maximum of one label set for breakpoint). *@rfc* — Run function call after which to interrupt the run (maximum of one run function call set for breakpoint). *v* — Variable name after which to interrupt the run (maximum of one variable set for breakpoint).
Cv *vld*	Changes the value of *v* to the variable, constant, reserved word, literal data, or expression contained in *vld*. Use the same kind of expressions you use with the Change Variable (CHG) statement.
D	Deletes all breakpoints.
E *n*	Examines line *n* of the run control report.

continued

Table 7-12. RDB Commands and Function Keys (cont.)

Command/Function Key	Description
F1	Steps through the run by executing a line at a time, including any continuation line.
F2	Steps through the run by executing one statement at a time (use when multiple statements occur on one line).
F4	Continues the run normally until one of these occurs: - The run encounters a breakpoint or an RDB statement - The run completes - The run aborts
M*v*	Monitors the contents of variable *v* whenever the value changes.
P[*reswd*]	Displays the contents of the specified reserved word. (You can also use this command to display the contents of named variables, as with the V command.) ***reswd*** Reserved word. Hyphens (-) on either side of the displayed contents denote the boundaries of the variable so that you can see significant characters and the justification of the contents within the variable (for example, - 0-). If you do not specify a parameter, the function displays the contents of all initialized variables on the next statement to execute.
R	Continues the run (same as **F4**).
S[*nmbr-of-lns*]	Sets streaming mode, which means that the run executes one logic line (or one statement if you had been pressing **F2**) at a time in a scrolling, uninterrupted manner. To stop streaming mode, press **Abort**. ***nmbr-of-lns*** Number of lines used, as a window at the bottom of your screen, for scrolling your run as it executes. The upper portion of the screen remains stationary.

Table 7-12. RDB Commands and Function Keys (cont.)

Command/Function Key	Description
U	Displays the run control report for updating (the run control report becomes the current -0).
V*n*	Displays the contents of the numbered variable ***n*** or the named variable ***nmvar*** (include < and > in ***nmvar***). Hyphens (-) on either side of the displayed contents denote the boundaries of the variable so that you can see significant characters and the justification of the contents within the variable (for example **- 0-**). To display the contents of named variables, you can use the P command or type the named variable.
W[*,c,d,r,l*|*,-n,l*]	Displays a window (10 lines) of a report or result beginning at a specified line. ***c,d,r*** Report to display. Default = run control report. ***l*** First line to display. Default = line 1. ***-n*** Result to display.
@rfc	Executes any run statement you specify (instead of executing statements in the run control report). For example, **@ldv v099=123 .** loads v099 with 123. Processing continues on the next line. ***rfc*** Run function call.
^	Exits RDB mode. If in streaming mode, press **Abort** first, then type **^** and transmit.

Comments

- When you start the RDB utility, it displays line 3 of your run control report and the RDB prompt. Your run is now under control of the RDB utility.
- When the run encounters an error, it stops and displays a system message but leaves you in RDB mode. All variables and results are available for you to examine.
- When the run encounters a statement that displays output, such as an Output (OUT) or Display Report (DSP) statement, the RDB utility repaints the previous output screen and displays the current statement.
- When the run encounters a Run Start (RUN) statement, the RDB utility terminates. To continue the RDB utility, the run started via the RUN statement must contain an RDB statement.

Examples

After requesting a display of the contents of v3 (space, space, 1, 5):

```
      8   @lzr,0,v2,1 v3i4 .
      9   @lok,0,v2,1 .
RDB♦ v3
-  15-
```

After requesting a display of the contents of <drawer> (B):

```
     8   @lzr,0,<drawer>,1 v3i4 .
RDB♦ <drawer>
-B-
```

After requesting the contents of v2 be changed to d:

```
     8   @lzr,0,v2,1 v3i4 .
RDB♦ c v2 d
```

Examine line 42 of your run control report:

```
     8   @lzr,0,v2,1 v3i4 .
RDB♦ e 42
```

Line 42 is displayed. Transmit again to display line 43. Press **F1** or **F2** to continue the run at the statement displayed before you used the E command.

Display a window of your run control report (four lines preceding and five lines following the current line):

```
RDB♦ w
```

Display a 10-line window starting at line 1 of the -0 result:

```
RDB♦ w,-0
```

RDB (Run Debug): OS 1100

This section applies only to the OS 1100 MAPPER System. See "RDB (Run Debug)" in this section for a description of the RDB statement as it applies to other MAPPER systems.

The Run Debug (RDB) statement helps you locate and correct any errors in a run while it is executing. Use the RDB statement in a run instead of the manual function request to start the debugging at a specific point in a run. This is especially useful for debugging large or complex runs.

With the Run Debug (RDB) utility, you can step through your run to do the following:

- Display the contents of a variable or reserved word; display renamed reports and results.
- Pause at a specified variable, run statement, line, or label.
- Make corrections to the run control report.
- Dynamically execute run statements, for example, to change the contents of a variable.

Condition

You must be registered as a run designer and you must be the one who last updated the run control report; otherwise the system ignores the RDB request.

Format

```
@RDB .
```

Manual Function Request

```
RDB run[,v,...v]
```

where *run* is the name of the run to debug and *v,...v* is data supplied to the run to be initialized via INPUT$.

RDB Commands and Function Keys

Use RDB commands and function keys **F1** through **F4** to view different kinds of information in your run. Table 7-13 describes the RDB commands and function keys.

Table 7-13. RDB Commands and Function Keys: OS 1100

Command/Function Key	Description	
B *prm*	Stops the run at the point (breakpoint) specified in *prm*. Only one breakpoint can be active at a time, and the previous breakpoint is cleared when you set a new one. Enter one of the following commands or **B** to clear the breakpoint:	
	B *n*	Line number at which to interrupt the run. The run stops before executing line *n*.
	B L*lab*	Letter L followed by the label number at which to interrupt the run. The run stops before executing the line at label *lab*.
	B *v*	Variable at which to interrupt the run. The run stops after executing any statement that refers to variable *v* (variable name or number).
	B *@rfc*	Run function call at which to interrupt the run. The run stops before executing run function call *rfc*.
F1	Steps through the run by executing a line at a time, including any continuation line.	
F2	Steps through the run by executing one statement at a time (use when multiple statements occur on one line). Note that the utility considers the period-space terminator a run statement.	
F3	Returns the run to normal execution speed. To stop the run, press **F3** a second time.	
F4	Sets streaming mode, which means that the run executes one logic line (or statement, if you had been pressing **F2**) at a time in a scrolling, uninterrupted manner. To stop the run, press **F4** a second time.	

continued

Table 7-13. RDB Commands and Function Keys: OS 1100 (cont.)

Command/Function Key	Description
M {*v*|**$**}	Monitors a variable or reserved word by displaying its contents whenever the run stops (between line steps or statement steps). Only one item can be monitored at a time, and the previous monitor is cleared when you set a new one. Enter one of the following commands or **M** to clear the monitor: **M** *v* — Variable to monitor. The size, type, and contents of variable *v* (variable name or number) are displayed whenever the run stops. **M** *reswd$* — Name of the reserved word to monitor (for example, USER$ or LLP$). The contents of *reswd$* are displayed whenever the run stops.
-n	Displays renamed report or result *n*.
R	Displays your run control report. You can manipulate this report or result on the screen (for example, roll or shift it) and you can update it with a line change or SOE update. However, you cannot use manual functions such as Search (S) or Locate (LOC). Press **Resume** to return to the run.
reswd$	Displays the name of the reserved word you specify (for example, USER$ or LLP$), followed by its contents enclosed in slant characters.
v	Displays the variable size and type of variable *v*, followed by its contents enclosed in slant (/) characters. You can enter a variable name or number, for example, <data> or v3. You do not need to enter the less than and greater than (< and >) symbols to display a named variable.

Table 7-13. RDB Commands and Function Keys: OS 1100 (cont.)

Command/Function Key	Description
@rfc	Dynamically inserts a temporary run statement to execute. Processing continues uninterrupted on the current line. *@rfc* Run function call followed by the remaining fields needed in the statement. For example, **@ldv v099=123 .** loads v099 with 123.
?	Displays a summary of RDB commands. Press **Resume** to return to the run.
^	Exits RDB mode, terminates the run, and releases the screen.

Comments

- When you start the RDB utility, it displays the RDB prompt followed by a line containing the name and location of the run. Below this information, line 3 of your run control report appears (up to 70 characters of the line on an 80-character screen; up to 122 characters on a 132-character screen).

  Note: *If you request the RDB utility using the RDB run statement, the utility displays the first statement following the RDB statement.*

  Your run is now under control of the RDB utility. The next statement to execute is displayed at the lowest displayed line on your screen.

- When the run encounters an error, press **Resume** to activate RDB error mode. In error mode, you can examine all variables, reserved words, and renamed reports and results as they existed when the run erred. While in error mode, only the RDB commands that display information (*-n*, R, V*v*, *reswd*$) are allowed. See Table 7-13.

- When the run encounters a statement that displays output, such as an Output (OUT) or Display Report (DSP) statement, the RDB utility displays the screen as requested. When you resume the run, it clears the screen and displays the RDB prompt.

- When the run encounters a Run Start (RUN) statement, if you were the last person to update the newly started run, the RDB utility continues to be active in that run.
- When the run executes a Defer Updates (DFU) statement, the RDB utility skips to either the Commit Updates (CMU) or Decommit Updates (DCU) statement, whichever comes first. The system executes any intervening statements and displays them in streaming mode, that is, in a scrolling, uninterrupted manner.

Examples

After requesting a display of the contents of v3 (RDB prompt line request was RDB♦v3):

```
RDB♦
RUN=RUNIT  REPORT=10F0

     5 ♦ @lzr,0,v2,1 v3i4 .
     6 ♦ @lok,0,v2,1 .
V3I4=/  31/
```

After requesting a display of the contents of <drawer> (RDB prompt line request was RDB♦<drawer>):

```
RDB♦
RUN=RUNIT  REPORT=10F0

     5 ♦ @lzr,0,<drawer>,1 v3i4 .
     6 ♦ @lok,0,<drawer>,1 .
<DRAWER>H1=/b/
```

After requesting a run statement be dynamically inserted (RDB prompt line request was RDB♦@ldv v2=d):

```
RDB♦
RUN=RUNIT  REPORT=10F0

     5 ♦ @lzr,0,v2,1 v3i4 .
INSERT-♦ @ldv v2=d
     5 ♦ @lzr,0,v2,1 v3i4 .
```

After requesting that v3 be monitored (RDB prompt line request was RDB♦m v3):

```
RDB♦
RUN=RUNIT  REPORT=10F0  MONITOR=V3

     5 ♦ @lzr,0,v2,1 v3i4 .
V3 - UNDEFINED
V3I4=/  31/
     6 ♦ @lok,0,v2,1 .
```

RDC (Read Continuous)

The Read Continuous (RDC) statement reads report or result lines or line segments.

Format

@RDC,*c*,*d*,*r*[,*l*,*q*,*ltyp*,*lab*] *cc vdata* .

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report from which lines are to be read.
l	Line number at which to start reading. Default = line 1.
q	Number of lines to read. Default = all lines.
ltyp	Line type to process (leave this blank if all types are to be read).
lab	Label to go to if no lines exist.
cc	Column-character positions or names of the fields to be read.
vdata	Variables (up to 40) to capture the data. To initialize a variable to the size of a field, specify only the variable name and type (for example, v1h, v2s, and so on).

Comments

- You must follow the RDC statement with an output line because an RDC statement does the following in sequence:
  - Reads a line
  - Places the data read in variables
  - Writes the data to the output area
  - Reads the next line
- Use only one line in the output area for the variables to be written by the RDC statement.
- The column-character fields do not have to be specified from lower-numbered to higher-numbered columns.

- An RDC statement generates one line in the output area for each line it reads from the input report.
- You can define the size of a variable to load in an RDC statement or you can define the variable to match the size of a corresponding report field. This is especially useful with a named field, because the name does not directly specify the field size; it also allows the RDC statement to adjust to a change in the size of a field.

Example 1: Reading All Lines

Read all lines of the current result, capture the data in v1, and include the variable in the output:

```
@rdc,-0 1-256 v1s256 .
v1
```

Example 2: Reading Data from Two Fields

Read data from two fields and capture the data in v1 and v2:

```
@rdc,0,b,2 'custcode','shipdate' v1h,v2i .
 v1 v2
```

where:

`0,b,2`	Reads lines in report 2B0
`'custcode', 'shipdate'`	Reads data from the Cust Code and Ship Date fields
`v1h`	Initializes v1 as a type H variable to the size of the Cust Code field to capture the data read from that field
`v2i`	Initializes v2 as a type I variable to the size of the Ship Date field to capture the data read from that field

Example 3: Reading Data and Placing Under Headings

Read data from two fields, capture it in v1 and v2, and place it in the output area under the headings of SHIPPING ORDER and DATE LAST STATUS CHANGE:

```
 SHIPPING ORDER      DATE LAST STATUS CHANGE

@rdc,0,b,2,6,50,□,099 'status','shiporder' v1i,v2h .
  v2                        v1
```

where:

`0,b,2`	Reads lines in report 2B0
`6`	Starts reading at line 6
`50,□`	Scans 50 lines, but reads tab lines only
`099`	Goes to label 99 if line 50 or the report does not exist
`'status'`	Reads data from the Status Date field (column 5 for six characters)
`'shiporder'`	Reads data from the Ship Order field (column 71 for five characters)
`v1i`	Initializes v1 as a type I variable to the size of the Status Date field to capture the data read from the field
`v2h`	Initializes v2 as a type H variable to the size of the Ship Order field to capture the data read from the field
`v2 v1`	Places the data in the output area in this format under the headings SHIPPING ORDER and DATE LAST STATUS CHANGE

Example 4: Reading Data and Reformatting

Read the drawer D heading lines from report 0D0, read data from report 2B0, and reformat it in the output area:

```
@brk,0,d .
@rdc,0,d,0,18,4 1-132 v1s132 .
v1
@rdc,0,b,2,6 1-80 v1s80 .
v1(1-3)v1(38-6) v1(14-11)    v1(44-6)
@brk .
```

RDL (Read Line)

The Read Line (RDL) statement reads a report or result line or segments of a line.

Format

@RDL,*c*,*d*,*r*,*l*[,*lab*] *cc vdata* .

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report from which to read the line.
l	Line number to read.
lab	Label to go to if the line or report does not exist.
cc	Column-character positions or names of the fields to be read.
vdata	Variables used to capture data (up to 40).
	When you load variables of a specified size, the size of the variable determines how many columns are read. For example, in the following statement, v1 contains a value of the line starting in column 5 for 10 positions: **@rdl,0,b,2,v10,099 5-6 v1h10** To define a variable to the size of a field, specify only the variable name and type (for example, v1h, v2s, and so on).

Reserved Word

LINE$ contains the next line number to read. If the last line read was beyond the end of the report, LINE$ contains a zero.

Comments

- The column-character fields do not have to be specified from lower-numbered to higher-numbered columns.
- If you need to read a series of lines in the same report, read the first line using the RDL statement; read subsequent lines using the Read Line Next (RLN) statement. See RLN in this section.
- You can define the size of a variable to load in an RDL statement or you can define the variable to match the size of a corresponding report field. This is especially useful with a named field, because the name does not directly specify the field size; it also allows the RDL statement to adjust to a change in the size of a field.
- See also RDC and RLN in this section.

Example 1: Reading Lines from a Field

Read lines from one field in report 2B0:

```
@rdl,0,b,2,6 'custcode' v1h .
```

where:

`0,b,2`	Reads lines in report 2B0
`6`	Reads line 6
`'custcode'`	Reads data from the Cust Code field
`v1h`	Initializes v1 as a type H variable to the size of the Cust Code field to capture the data read from that field

Example 2: Reading Data from Two Fields

Read lines from two fields in report 2B0:

```
@rdl,0,b,2,<line>,099 'status','shiporder' \
<status>i,<ord>h .
```

or

```
@rdl,0,b,2,<line>,099 5-6,71-5 <status>i,<ord>h .
```

where:

`0,b,2`	Reads a line in report 2B0
`<line>`	Reads the line number in <line>
`099`	Goes to label 99 if the line or the report does not exist
`'status'` `5-6`	Reads data from the Status Date field (column 5 for six characters)
`'shiporder'` `71-5`	Reads data from the Ship Order field (column 71 for five characters)
`<status>i`	Initializes <status> as a type I variable to the size of the Status Date field to capture the data read from the field
`<ord>h`	Initializes <ord> as a type H variable to the size of the Ship Order field to capture the data read from the field

REH (Retrieve from History)

This section applies only to the OS 1100 MAPPER System.

The Retrieve from History (REH) statement retrieves the version of a MAPPER report prior to the last system purge or merge process.

Format

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

Following is a description of the field and subfields:

Field	Description
c,d,r	Report to retrieve.
lab	Label to go to if an error occurs.

Reserved Word

If the run continues at the label, STAT1$ contains an error code indicating the reason (if any) why the data cannot be retrieved by the system. If you do not use the *lab* subfield and an error occurs, the run errs.

Error codes:

4 Drawer was added since last purge (prior version does not exist).

13 Prior version does not exist.

16 Internal software error.

17 Specified MAPPER file is being merged.

Example

Retrieve report 2A0, and go to label 99 in case of error:

```
@reh,0,a,2,099 .
```

RER (Register Error Routine)

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

Since all variables, temporary results, and the current output area remain unaltered and are accessible to the error routine, use the routine to help debug the run.

Format

```
@RER[,c,d,r lab | lab] .
```

Following is a description of the field and subfields:

Field	Description
c,d,r	Report containing the external routine. **1100:** The report containing the external routine must be in the same character set type as the calling run control report.
lab	Label in the report where the error routine begins. To begin the routine at the first line of the external run control report, use LIN1 in this field.

Reserved Words

The following reserved words contain zero until a run errs. The system resets them to zero whenever the run refers to CERR$ or executes a noninterim display with a Display Graphics (DSG), Display Message (DSM), Display Report (DSP), Output (OUT), or Screen Control (SC) statement.

1100: On OS 1100 MAPPER Systems, the system resets them to zero whenever the run executes a noninterim display with the DSG, DSM, DSP, OUT, or SC statements.

AXDRW$ contains the alphabetic drawer of the run control report where the run erred.

XDRW$ contains the numeric drawer of the run control report where the run erred.

XERR$ contains the message number of the error — use this number to retrieve the message with a Load System Message (LSM) statement.

XFUN$ contains the failing statement call when the run erred.

XLINE$ contains the line number in the report where the run erred.

XRPT$ contains the report number of the run control report where the run erred.

Comments

- If an error occurs, the system runs the registered error routine and cancels any previously registered abort and error routines and update locks.
- Place the RER statement as close as possible to the beginning of the run. If an error occurs before the run executes the RER statement, the run has no error routine to go to.
- The system cancels the error routine when the run terminates, but you can also use the Clear Error Routine (CER) statement to cancel the routine.
- The error routine cannot contain the statements GTO RPX, RAR, RER, or RUN. You can use an RSR statement only if it starts an internal subroutine.
- If an error occurs while an error routine is executing, the run terminates with a normal system message.
- I/O and LLP limits and cabinet restrictions from the run also apply to the error routine.
- A noninterim display specified by any of the following statements in your error routine continues normal run processing:

  DSG, DSM, DSP, OUT, SC

- Any reference to the reserved word CERR$ sets CERR$ and XERR$ to zero and continues normal run processing.

1100: On OS 1100 MAPPER Systems, any reference to CERR$ reserved word clears CERR$ and continues normal run processing.

- The values of reserved words IO$ and LLP$ reflect the number of I/Os and LLPs used in the error routine and the run.

1100: On OS 1100 MAPPER Systems, IO$ and LLP$ reserved words in an error routine reflect the I/Os and LLPs in the error routine itself.

- Error routines registered in called routines (via the CALL statement) override previously registered error routines; however, when control returns to the calling run, the system automatically reregisters the original error routine.

1100: On OS 1100 MAPPER Systems, the system does not automatically reregister the original error routine; you must reregister it yourself.

- If you do not register an error routine in the called routine, in case of an error, control goes to the error routine in the calling run.

1100: On OS 1100 MAPPER Systems, control does not go to any error routines previously registered in the calling run.

- **1100:** Use the RUNERR error subroutine as a debugging tool during development of your runs or as an error notification routine in production runs. See the online help system (HELP,@RER) for more information.

Example

This statement registers the external error routine at label 5 in report 2E0:

```
@rer,0,e,2 005 .
```

RET (Retrieve File)

This section does not apply to the OS 1100 MAPPER System. See "RET (Retrieve File): OS 1100" in this section for a description of the RET statement as it applies to the OS 1100 MAPPER System.

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

Format

```
@RET,c,d[,mapperf?,hdgs?,lab] fn .
```

Following is a description of the field and subfields:

Field	Description
c,d	Cabinet and drawer into which the report should be placed.
mapperf?	File is in MAPPER format, Y or N. Default = N. If file is in binary format, enter B.
hdgs?	Add headings from the receiving drawer to the result, Y or N. Default = N.
lab	Label to go to if the file is not found.
fn	Name of the file to retrieve. **U Series and UNIX:** Include the full path name. File names can be uppercase or lowercase but must exist exactly as typed. For example, Abc must be accessed with an uppercase A and lowercase b and c. **A Series:** Include a usercode and pack family (optional). File names must be uppercase. **BTOS:** Use the full path name: [*volume*]<*directory*>*filename*. File names are not case sensitive. **PC MAPPER:** Include the complete path name. The directory you specify for the file must exist. If you use a reverse slant (\ ) to define a directory path in an RET statement, it must be enclosed in apostrophes. This prevents the directory path from being misinterpreted as a MAPPER run statement that continues on the next line.

Comments

- If the line length of the drawer into which you are retrieving a MAPPER-formatted file is less than that of the file, any extra characters in the file are truncated. If the file is not in MAPPER format, extra characters are wrapped to the next line.
- If you created a file with headings and retrieved it with headings, the result contains two sets of headings. Conversely, if you create a file with no headings and retrieved it with no headings, the result contains no headings, only the date line.
- When you retrieve a file that is not in MAPPER format, the RET statement always adds the date line to the result. A report created *with* headings and *not* in MAPPER format always contains two date lines when you retrieve the file.

Example

PC MAPPER: The following example does not apply to the Personal Computer MAPPER System.

Retrieve a file that is not in MAPPER format:

```
@ret,0,b,n,n WORK/PROD/STATUS .
```

The file WORK/PROD/STATUS is created as a result in cabinet 0, drawer B, with no headings.

PC MAPPER: The following example applies only to the Personal Computer MAPPER System.

This statement retrieves the status.new file in the \work\prod directory, which is not in MAPPER format. The statement does not add headings to the cabinet 0, drawer B result.

```
@ret,0,b,n,n '\work\prod\status.new' .
```

RET (Retrieve File): OS 1100

This section applies only to the OS 1100 MAPPER System. See "RET (Retrieve File)" in this section for a description of the RET statement as it applies to other MAPPER systems.

The Retrieve File (RET) statement retrieves OS 1100 program or data files. The retrieved file becomes the -0 result.

Note: *When bringing large volumes of data from the batch environment, retrieve the data with a RET statement instead of sending it through the batch port.*

Condition

The file must be a sector-formatted file with no read or write keys.

Format

```
@RET,c,d[,lab] [qual],fn[,cyc,elt,ver,mapperf?,hdgs?,l,ststr,q] .
```

Following are descriptions of the fields and subfields:

Field	Description
c,d	Cabinet and drawer into which the report should be placed.
lab	Label to go to in case the file cannot be retrieved.
qual	Qualifier.
fn	Name of the file to retrieve.
cyc	Relative or absolute file cycle number.
elt	Element name. Leave this field blank for a data file.
ver	Version.
mapperf?	File is in MAPPER format? Y or N. Default = N. (A Y here overrides a Y for *hdgs?*).
hdgs?	Add headings from the receiving drawer to the result, Y or N. Default = N.
l	Line number at which to start the retrieval.
ststr	Character string to locate. The retrieval starts on the line that contains this character string.
q	Number of lines to retrieve.

Reserved Words

STAT1$ contains the following status codes if the statement is not able to retrieve the file:

1 Insufficient data.
2 Improperly formatted name (such as an illegal character).
3 Facility reject.
4 File does not exist.
5 File rolled out.
6 File exclusively assigned to another run.
7 Facilities currently unavailable.
8 Private file, under different project-id.
9 File may not be read (write only or missing key).
10 File is not sector-formatted mass storage file.
11 File is not program file (if element specified).
12 Element requested does not exist.
13 Report requested did not exist yesterday.
14 File is MAPPER file: Use the manual Retrieve File (RET) function.
15 Specified MAPPER file is being merged.
16 Internal software error.
17 Cannot retrieve report while file is being merged.
18 Locate string is not found.
19 Program file must include element name.
20 File exists but is not readable.

Example

Retrieves the file myqual*myfile starting at line 100:

```
@ret,0,a,099 myqual,myfile,,,,,,100 .
```

where:

0,a	Places the result in cabinet 0, drawer A.
099	Goes to label 99 in case of error.
myqual	Retrieves a file with a qualifier of MYQUAL.
myfile	Retrieves the file named MYFILE.
100	Starts retrieving at line 100.

RFM (Reformat Report)

The Reformat Report (RFM) statement moves columns of data from a report to a newly formatted report or result containing only heading lines. The RFM statement creates a result containing the same line types and number of lines as the issuing report.

Format

@RFM,*ic,id,ir,rc,rd,rr o icc ,ip rcc ,rp* .

Following is a description of the fields and subfields:

Field	Description
ic,id,ir	Issuing report.
rc,rd,rr	Receiving report.
o	Options field. Designate no options (' '). No options are available.
icc	Column-character positions or names of the fields of the data to be moved in the issuing report.
ip	Parameters of the issuing report data (A-M).  **1100:** On OS 1100 MAPPER Systems, you can use parameters A through Z.
rcc	Column-character positions or names of the fields in the receiving report.
rp	Parameters of the receiving report data (A-M).  **1100:** On OS 1100 MAPPER Systems, you can use parameters A through Z.

Comments

- The RFM statement moves the columns specified and the line type designator from the issuing report to the receiving report. If the issuing report contains any period type lines, the entire period type line is moved to the receiving report.
- The RFM statement processes all line types.
- Note that you can also use a Read Continuous (RDC) statement to reformat a report using the output area of the run (see RDC). Use the Match (MCH) statement if you want to conditionally move and extract data from two different reports or between a displayed result and a specified report, where both contain matching data (see MCH).

Example

Move data from report 2B0 to report 2D0:

```
@rfm,0,b,2,0,d,2 '' 'product' ,a 'product' ,a .
```

or

```
@rfm,0,b,2,0,d,2 '' 15-9 ,a 12-9 ,a .
```

where:

`0,b,2` `0,d,2`	Moves data from report 2B0 to report 2D0

`'product'`
`15-9`
`a`
	Moves line type and data from the Product Type field (column 15 for nine characters) in report 2B0

`'product'`
`12-9`
`a`
	to the Product Type field (column 12 for nine characters) in report 2D0

RLN (Read Line Next)

The Read Line Next (RLN) statement continues reading from the report or current result from which a Read Line (RDL) or Find and Read Line (FDR) statement was just executed.

Condition

An RLN statement must follow a successful FDR or RDL statement.

Format

```
@RLN[,l,lab] cc vdata .
```

Following is a description of the fields and subfields:

Field	Description
l	Line number to read. Default = next line number.
lab	Label to go to if the specified line does not exist.
cc	Column-character positions or names of the fields to read. You do not have to specify fields in order from left to right, but be sure to assign the variables that capture the data in matching sequence.
vdata	Variables (up to 40) to capture the data. To initialize a variable to the size of a field, specify only the variable name and type (for example, v1h, v2s, and so on).

Reserved Word

LINE$ contains the next line number to read. If the line read is beyond the end of the report, LINE$ contains a zero.

Comments

- Since the RLN statement continues processing a previously specified report, the only statements allowed between RDL and RLN (or FDR and RLN) statements are those that do not end the processing on the report, such as IF, CHG, and LDV.
- When most other statements, such as those with a manual function counterpart, begin processing, they signal to MAPPER software that the run has finished processing the previous report. Some of these statements include ART, DC, DEV, FND, LZR, MCH, SRH, SOR, TOT, and WRL.
- When you load variables of a specified size, the size of each variable determines the number of columns the statement reads.

  For example, in the following statement <data> contains ten characters of data starting with column 5, even though you specified 2 characters in the *cc* field:

  ```
  @rln,,099 5-2 <data>h10 .
  ```

- To have the run automatically initialize a variable to the size of a field, specify only the variable name and type (for example, <data1h>, <data2s>, and so on). The RLN statement initializes the variables the appropriate size for the fields it reads.

Example 1: Reading Data from a Field

Read data from the Cust Code field and capture the data in v1:

```
@rln 'custcode' v1h .
```

where:

`'custcode'`	Reads data from the Cust Code field
`v1h`	Initializes v1 as a type H variable to the size of the Cust Code field to capture the data read from that field.

Example 2: Reading Data from Two Fields

Read data from two fields in report 2B0:

```
@rdl,0,b,2,<line>,099 'status','shiporder' \
<status>i,<ord>h .
  .
  .
  .
@rln,,099 'status','shiporder' <status>,<ord> .
```

where:

`0,b,2`	Reads a line in report 2B0
`<line>`	Reads the line number in <line> (the following RLN statement automatically reads the line in <line> + 1)
`099`	Goes to label 99 if the line or the report does not exist
`'status'`	Reads data from the Status Date field
`'shiporder'`	Reads data from the Ship Order field
`<status>i`	Initializes <status> as a type I variable to the size of the Status Date field to capture the data read from the field
`<ord>h`	Initializes <ord> as a type H variable to the size of the Ship Order field to capture the data read from the field

RPW (Read Password)

The Read Password (RPW) statement assigns up to two read passwords to a run. Use the RPW statement to access reports with read passwords that you plan to process in a run.

Format

```
@RPW{ ,pw1 | ,pw1 ,pw2 | , ,pw2 }  .
```

Following is a description of the subfields:

Field	Description
pw1	First read password.
pw2	Second read password.

Comments

- You can assign different passwords to access different reports many times in the run. For example, this statement assigns two read passwords to a run:

  ```
  @rpw,xxx,yyy .
  ```

- This statement changes the second read password (from the previous example) so the run can access a different report. The first read password remains assigned:

  ```
  @rpw,,zzz .
  ```

Example 1: Using One Read Password

Use a read password of readme:

```
@rpw,readme .
@srh,0,b,2 '' 'custcode' □,amco .
```

Example 2: Using Two Read Passwords

Use two read passwords (this is useful, for example, in a statement that must access two reports):

```
@rpw,readme,readit .
@mch,0,b,2,0,c,1 n 'product' □,1 .
```

Example 3: Capturing the Password with INPUT$

For greater security, solicit the password from a user and conceal it in the run:

```
@chg input$ v1h6 .
@rpw,v1 .
```

RS (Run Status)

This section applies only to the OS 1100 MAPPER System.

The Run Status (RS) statement creates a -0 result listing the status of runs being executed with your user-id.

Format

```
@RS[,o,run,sn] .
```

Following is a description of the subfields:

Field	Description
o	Options field (see Options).
run	Run name (leave blank for the status of all runs).
sn	Station number (for the status of only those runs at this station).

Options

Blank	All active and suspended runs
A	Active runs only
B	Background runs only
S	Suspended runs only

Examples

Obtaining the status of all runs:

```
@rs .
```

Obtaining the status of the background run myrun:

```
@rs,b,myrun .
```

Obtaining the status of all runs at station 123:

```
@rs,,,123 .
```

RSI (Remote Symbiont Interface)

This section applies only to the OS 1100 MAPPER System.

The Remote Symbiont Interface (RSI) statement initiates an interactive demand program through a MAPPER run.

When a MAPPER run calls the RSI function, the run terminates; then the MAPPER system signs the user on to the operating system in demand mode, and gives the user manual control of the display terminal.

Note: *The RSI interface of the Exec is text oriented. Therefore, the MAPPER system cannot determine whether the parameters in the RSI statement are causing an error. For example, an invalid user-id, password, or a nonexistent ADD file may cause an error.*

Condition

If the MAPPER software user is not a valid RSI user (as defined in the user registration report), no @ or @@ commands (except for @@TERM and @@X) are allowed.

Format

@RSI[,*site-id*,*tmo*] *user*,*pw*[,*acct*,*qual*,*fn*,*cyc*,*elt*,*ver*,*q*] .

Following is a description of the subfields:

Field	Description
site-id	Site identifier (six characters) for this display terminal.
tmo	Number of seconds to wait for demand input at the terminal. If left blank, no timeouts occur. Pressing **Abort** before receiving manual control aborts the RSI statement.
user	Demand mode user-id.
pw	Demand mode password that validates the user-id.

continued

continued

Field	Description
acct	Account number. If specified, the MAPPER system submits an executive control language (ECL) RUN statement in this format: **@RUN** *run-id,acct/,proj-id* *run-id* Active site-id. *acct* Account number. *proj-id* Qualifier (*qual* from the RSI statement). If you did not specify *qual*, this field is left blank.
qual	File name qualifier.
fn	File name. If specified, the MAPPER system submits an ECL ADD statement after demand mode sign-on is complete. If omitted from the RSI statement, the following file-related subfields are left blank.
cyc	File cycle number.
elt	Element name.
ver	Element version name.
q	Number of text lines to display before giving manual control to the user. Default = 1 (minimum). Maximum = one less than the vertical screen size of the display terminal. If a demand program executes a control command that the Communications Control Routine (CCR) for the MAPPER system interprets before it solicits input from the terminal, it discards everything that preceded the control command.

RSR (Run Subroutine)

The Run Subroutine (RSR) statement executes an external or internal subroutine starting at a specified label.

Format

```
@RSR{,c,d,r lab | lab} .
```

Following is a description of the field and subfields:

Field	Description
c,d,r	Report containing the external subroutine. **1100:** The report containing the external routine must be in the same character set type as the calling run control report.
lab	Label where the subroutine starts. This field is required. To begin the routine at the first line of the external run control report, use LIN1 in this field.

Reserved Words

Use the reserved words ACDRW$ and CRPT$ in an external subroutine to determine the drawer and report number of the calling run control report.

Comments

- To save the contents of all currently defined variables and pass control to a subroutine, use the Call Subroutine (CALL) statement. You can pass variables to the subroutine, manipulate them, and pass them back to the calling run without affecting any of the other currently defined variables.
- You can refer to all variables and results in the calling run from subroutines. When you return to the calling run, newly created variables and results in the subroutines become available. However, if in the subroutine you reinitialize variables or rename results from the calling run, the subroutine overwrites them.
- You can nest subroutines in a run. Internal subroutines can call subroutines that call other subroutines up to 10 levels. However, you can call only one external subroutine.

- Use an End Subroutine (ESR) statement to exit the subroutine and return control to a specified line in the run control report relative to the line containing the calling RSR statement.
- Use a Clear Subroutine (CSR) statement in an external subroutine to execute another external subroutine. A CSR statement allows an external subroutine to call another external subroutine but does not allow a return to the original calling run control report with an ESR statement.
- When registering your run, inform the MAPPER system coordinator of any other run control reports containing external subroutines.
- Do not place other run statements on the same line after the RSR statement. Other run statements following it on the same line are ignored. Put the next run statement on a new line.

Example 1: Using an Internal Subroutine

Use an RSR statement to execute the subroutine at label 1 (when the subroutine ends, it returns to the calling RSR+1 line, which is *run statement a*):

```
@    rsr 001 .
@ . run statement a .
@ . run statement b .
@ . run statement c .
@    gto end .
@001: subroutine run statement a .
@ . subroutine run statement b .
@ . subroutine run statement c .
@    esr .
```

Example 2: Using an External Subroutine

Use an RSR statement to execute an external subroutine in report 2E0, starting at label 2:

```
@rsr,0,e,2 002 .
```

Example 3: Nesting Subroutines

This example uses an RSR statement to execute the subroutine at label 2. That subroutine uses another RSR statement to execute a subroutine at label 3. When the second subroutine completes, it uses an ESR statement to return to the second part of the first subroutine. When that completes, it uses another ESR statement to return to the original run statement (*run statement a*):

```
@     rsr 002 .
@001: run statement a .
@ . run statement b .
@ . run statement c .
@     GTO END .
@002: first subroutine run statement a .
@ . first subroutine run statement b .
@ . first subroutine run statement c .
@     RSR 3 .
@ . second part of first subroutine d .
@ . second part of first subroutine e .
@ . second part of first subroutine f .
@     ESR .
@003: second subroutine run statement a .
@ . second subroutine run statement b .
@ . second subroutine run statement c .
@     ESR .
```

RUN (Run Start)

The Run Start (RUN) statement terminates the current run and starts another run at the same MAPPER station.

Format

```
@RUN {run[,vld]|"cmd"} .
```

Following is a description of the field and subfield:

Field	Description
run	Name of the run to start.
vld	Variables, literal data, reserved words, or any combination of these, to transfer to the run. Use INPUT$ to initialize these variables.
"cmd"	Run name or manual function call to execute, including input parameters to be sent to the run or manual function. The quotation marks (") are required.
	1100: This field is not available in limited character set (LCS) run control reports.

Comments

- The manual equivalent of the RUN statement is to start a run by entering the name of the run.
- You can identify up to 40 variables to capture in the started run with INPUT$ (see Table 4-1 for maximum variable sizes). If you plan to use these variables as parameters in report processing functions such as Binary Find (BFN) or Search (SRH), you should use the Load Variable (LDV) statement with the P option to pack the variables (see LDV).
- Do not put other run statements on the same line after a RUN statement. The logic scan of the line terminates after executing the RUN statement.
- You can also start utility runs such as chart runs. See the online help system (HELP,@RUN) for details.

Example

Start the run named test and send the variables v1 and v2 and the character string SAM to that run (the run initializes these as variables with INPUT$). The current result (-0) is also available to the run:

```
@run test,v1,v2,SAM .
```

SC (Screen Control)

Use the Screen Control (SC) statement to create menus, input screens, and overlays using screen commands.

You can use screen commands to perform basic screen operations, which include positioning the cursor, clearing the screen, defining fields, and defining screen attributes. You can also use screen commands to design reports that define fields, box data in a specified area, and map function keys. See "Screen Commands" in this subsection for more information.

When the SC output is displayed, the run is suspended until the user presses **Transmit** or a function key, unless the statement options indicate an interim output or a forced transmit.

You can also design your screens using a menu-driven screen generator called the SCGEN run. For more information, see Section 5, or enter `apt` on the control line, tab to SCGEN, and press **Help**.

The SC statement has two formats. Use format 1 to directly specify screen commands, or use format 2 to access a report containing screen commands (also referred to as a form).

Condition

If you use format 2, a report containing executable screen commands must exist.

Formats

```
@SC[,,,,,,tabp,sn,lab] o scmnd .

@SC,c,d,r[,l,q,tabp,sn,lab] o [fldtxt] .
```

Format 1

@SC[,,,,,,*tabp*,*sn*,*lab*] *o scmnd* .

Following is a description of the fields and subfields:

Field	Description
tabp	Tab position at which to place the cursor, where a positive number is the number of tab positions forward (maximum of 100) from the home position and a negative number is the number of tab positions backward (maximum of 100) from the home position.
1100: ***sn***	Station number where output is to be displayed. If you omit this field or specify 0 (zero), output is displayed at the station executing the run.
1100: ***lab***	Label to go to if the output to another station cannot be completed successfully. If you omit this field and the run cannot be successfully completed, it is terminated with an error.
o	Options. See Options.
scmnd	Screen command or commands to be executed. See "Screen Commands" in this subsection.

Format 2

```
@SC,c,d,r[,l,q,tabp,sn,lab] o [fldtxt] .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report containing screen commands (also referred to as a form).
l	Line number in the specified report at which to start reading screen control commands. Default = 2.
q	Number of lines to read in the specified report. Default = all lines to the end of the report.
tabp	Tab position at which to place the cursor, where a positive number is the number of tab positions forward (maximum of 100) from the home position, and a negative number is the number of tab positions backward (maximum of 100) from the home position. Default = home position.
1100: *sn*	Station number where output is to be displayed. If you omit this field or specify 0 (zero), output is displayed at the station executing the run.
1100: *lab*	Label to go to if the output to another station cannot be completed successfully. If you omit this field and the run cannot be successfully completed, it is terminated with an error.
o	Options. See Options.
fldtxt	Field text — text to be inserted into unprotected fields (see the FLD and DFLD/AREA commands under "Screen Commands"). This field can contain a list of items separated by commas. The first item is placed in the first field, the second item in the second field, and so on. Note that this field is not interpreted unless the T (field text) option is specified in the SC statement.

Options

B Blink. Changes the less than (<) and greater than (>) signs to the left and right blink characters in all literal text. The blink characters may vary from terminal to terminal. See also the LB and RB commands in "Text Handling Commands" in this subsection.

C Center. Allows screen output (such as menus) designed to be centered on standard 80-character display terminals to be centered on terminals of other widths.

H Home Cursor Bypass. Bypasses the initial home cursor (HC) command that is automatically performed by the SC statement prior to executing the first screen command.

I Interim Output. Resumes the run automatically when the output is completed. Input is not accepted. See also the Q option.

K Key Mapping. Does not disable any current function key mapping. By default, the SC statement disables any currently mapped function keys prior to executing the first screen command.

L Line Control. Allows you to regain line control on the previously displayed report using the **Paint** key.

M Mask. Sends the data, but not spaces. (If data exists on a line, only the columns containing new data are overwritten. The old and new data are blended together on the same line.)

O Overlay menu. This option reprocesses the screen commands, but only repaints the text. It efficiently repaints or puts new data onto a menu that is already on the screen.

Q Quick Output. Resumes the run automatically when the output is displayed. This option is similar to the I option except that the run does not suspend, pending completion of the output, but rather continues as soon as the output is displayed.

S Sends the data including all spaces. (The default method of sending characters is that if fewer than six consecutive spaces appear, the system sends the data, including spaces. If six or more spaces appear, the cursor skips over the positions where the spaces would have been placed.)

T Field Text. Inserts text supplied in the *fldtxt* field into unprotected screen fields created with the FLD command. See the FLD and DFLD commands under "Screen Commands." Using this option, you can add values to menu fields. The T option is valid only when a report is specified in the SC statement (see Format 2).

U Update Control. Overlays a current report with data (for example, to display a message) but allows the user to perform manual SOE updates by holding update control of the report. This option is only meaningful when a report is on display.

X Causes a screen to be transmitted as it is displayed, just as if the user had pressed **Transmit**.

Reserved Words

Use reserved words to tailor screens to the site, user, and terminal. Using reserved words also enhances the portability of screens.

You can use the following reserved words with the SC statement. (See Appendix B for the contents of each word.)

- ADRW$
- AKEY$
- CAB1$
- CAB$
- DATE0$ through DATE8$
- DAY$
- DEPN$
- DEPT$
- DLINE$
- F1$ through F10$
- KKEY$
- LRRSD$
- MAPER$
- MKEY$
- RPT$
- STNUM$
- TIC$
- TIME$
- USER$
- XKEY$

The following reserved words are recognized only by the SC statement:

ALERT$ The current site alert message text.

DBASE$ The current site database size threshold message text.

1100: The DBASE$ reserved word is not available on OS 1100 MAPPER Systems.

1100: DMSG$ System down or purge time message text.

DUPLX$ The current site duplex file integrity error message text.

1100: The DUPLX$ reserved word is not available on OS 1100 MAPPER Systems.

LOGO$ Site-defined text for the sign-on and active screens.

STERR$ The current site startup error message text.

1100: The STERR$ reserved word is not available on OS 1100 MAPPER Systems.

Error Status Reserved Words

If the run errs while processing an SC statement, it loads STAT1$ and STAT2$ with the following information:

- STAT1$ contains the column number where the error occurred. If screen commands are being read from a report, STAT1$ contains a report column; otherwise, it contains the character position in the interpreted screen command string (from the *scmnd* field) of the SC statement.
- STAT2$ contains the report line number where the error occurred in the screen command report (format 2 only).

Screen Commands

There are six categories of screen commands. Each category comprises a unique set of commands; the commands are described in the subsections that follow.

- Cursor control commands
- Screen editing commands
- Field and attribute commands
- Text handling commands
- Screen printing commands

PC MAPPER: Screen printing commands are not available on Personal Computer MAPPER Systems.

- Setup commands

How Screen Commands Are Interpreted

- To provide a known starting point, the system automatically moves the cursor to the home position before the first screen command is interpreted.
- Screen commands are executed in the order they are specified.

- When commands are read from a report, a trailing space in any subfield or a space in column 1 terminates the scan of the current report line. You can use the remaining portion of the line for comments.
- Not all screen commands produce the same results on all types of display terminals. Whenever possible, screen commands that cannot be executed on a particular terminal are ignored.

How Data Is Interpreted

- By default, the SC statement disables any current function key mapping before executing the first screen command. Use the K option to keep the current function key mapping enabled.
- Any string of data that is not the name of a valid screen command is treated as literal text.

Syntax Rules for Screen Commands

- Use semicolons (;) to separate commands.
- If a screen command is too long to fit on one report line, you can continue it on the next line. Any command subfield that specifies text, such as field or message text, recognizes the reverse slant (\ ) as the continuation character when it occurs outside of literal delimiters.
- Follow these guidelines when specifying screen coordinates:
  - If you omit a value, the current cursor position is used.
  - If you specify a signed value (for example, +3 or -10), it is assumed to be relative to the current cursor position.
  - The home position is 1,1 (row 1, column 1).
- Enclose attribute parameters in parentheses. This is necessary because the attribute parameters together are actually one subfield. For example:

```
att,(rv,whi/red)

def,1,(pr,rv,tc);  att,1
```

See Using Inline Attributes, ATT Command and DEF Command under "Field and Attribute Commands" for more information.

- Literal text, such as field or message text, that contains spaces, commas, or semicolons must be enclosed in apostrophes (') or whatever literal delimiter is in use. See the TIC command under "Text Handling Commands." When using format 1 of the SC statement, the TIC$ reserved word is usually the easiest way to accomplish this.
- If you are building screen commands in a run output area, use a sequence of two apostrophes to produce one apostrophe in your data.
- Literal text that contains spaces must be enclosed in a double set of literal delimiters.

  This example statement uses format 1 and displays "Glen Ellyn." The delimiters are % and '. Note that the TIC command defines % as a literal delimiter.

```
@sc '' prep;tic,%;'%Glen Ellyn%' .
```

- When using format 2 of the SC statement, use a sequence of two apostrophes to produce one apostrophe in your data.

  When using format 1, define the literal delimiter to be a character other than an apostrophe and use the reserved word TIC$ to equal an apostrophe in the literal text. This example statement defines the delimiter as % and displays "We've".

```
@sc '' prep;tic,%;%We%tic$%ve% .
```

- To insert variables in literal text, follow the general guidelines set for literal text and place the variable name in the literal text.

  For example, this statement places the value of v2 in the literal phrase.

```
@sc '' prep;tic,%;'%Let %'v2'% out%'
```

Cursor Control Commands

Table 7-14 lists the commands that control the position of the cursor on the screen.

Table 7-14. Cursor Control Commands

Command	Description
HC	Home Cursor. Moves the cursor to the home position.
PC,*r*,*c*	Position Cursor. Moves the cursor to the specified row (*r*) and column (*c*) position. When specifying relative cursor coordinates, the cursor wraps around the edges of the screen as if you were moving the cursor manually.
CR[,*n*]	Cursor Return. Sends a cursor return to move the cursor to column 1 of the next screen line. Repeat *n* times. Default = 1.
TAB[,*n*]	Tab Cursor. Tabs the cursor forward *n* positions. Default = 1. If *n* is negative, the cursor tabs backward. Note that when you use the TAB command, the cursor position becomes unknown by the SC statement. Therefore, after you use the TAB command, you cannot use a command that references the current cursor position until you use a command that reestablishes the cursor location (such as HC, PC, FLD, or MSG).

Screen Editing Commands

Table 7-15 lists the commands that perform on-screen editing operations.

Table 7-15. Screen Editing Commands

Command	Description
CS	Clear Screen. Erases the entire display and moves the cursor to the home position.
ED	Erase Display. Erases the display from the current cursor position to the end of the screen.
EUD	Erase Unprotected Display. Erases the unprotected portion of the display from the current cursor position to the end of the screen. Attributes and protected fields are not erased.
PD	Protect Display. Protects the display beginning at the cursor position and continuing to the end of the display. Using PD does not alter the text and attributes that you are planning to display. You can also use PD to prepare the screen for overlaying.
DIL[,*n*]	Delete in Line. Deletes the unprotected character under the cursor, shifting succeeding characters, up to the end of the field or line (whichever occurs first), to the left. Repeat *n* times. Default = 1.
DID[,*n*]	Delete in Display. Deletes the unprotected character under the cursor, shifting succeeding characters, up to the end of the field or display (whichever occurs first), to the left. Repeat *n* times. Default = 1.
IIL[,*n*]	Insert in Line. Inserts a space at the current cursor position, shifting the character under the cursor and all succeeding characters up to the end of the field or line (whichever occurs first), to the right. Repeat *n* times. Default = 1.
IID[,*n*]	Insert in Display. Inserts a space at the current cursor position, shifting the character under the cursor and all succeeding characters up to the end of the field or display (whichever occurs first), to the right. Repeat *n* times. Default = 1.

continued

Table 7-15. Screen Editing Commands (cont.)

Command	Description
DL[,*n*]	Delete Line. Deletes the line the cursor is located on, moving the remaining lines up. Repeat *n* times. Default = 1.
IL[,*n*]	Insert Line. Inserts a blank line at the cursor location, moving the line the cursor is located on and all remaining lines down. Repeat *n* times. Default = 1.
DUP[,*n*]	Duplicate Line. Duplicates the line the cursor is on onto the line directly below it. The cursor is positioned on the duplicated line. Repeat *n* times. Default = 1.
EEL	Erase to End of Line. Erases from the current position to the end of the line or field, whichever occurs first. Attributes and protected fields are not erased.

Field and Attribute Commands

Use field and attribute commands to define fields, areas, and their attributes. You can place attributes anywhere on the screen. The attribute you define extends forward to the immediate left of the next attribute or to the end of the screen, whichever occurs first. See "Field Attribute Parameters" in this subsection for more information.

- Attribute (ATT)
- Define Attribute (DEF)
- Field (FLD)
- Message (MSG)
- Prepare Screen for Painting (PREP)
- Define Field (DFLD) and AREA

Attribute (ATT) Command

Use the ATT command to output an attribute at the current or designated cursor position.

Format

```
ATT,attr[,r,c]
```

Following is a description of the subfields:

Field	Description
attr	Attribute to be output. See "Field Attribute Parameters" in this subsection.
r,c	Cursor position where attribute begins. See Syntax Rules for Screen Commands under "Screen Commands" for rules on specifying attributes.

Example

This example positions the cursor to row 2, column 1, and outputs to the screen an attribute that has protected input and a color of white on blue:

```
att,(pr,whi/blu),2,1
```

Define Attribute (DEF) Command

Use the DEF command to define a numbered attribute for subsequent use in other screen commands. Once defined, you can use a numbered attribute anywhere an attribute is required.

Format

```
DEF,n[,attr]
```

Following is a description of the subfields:

Field	Description
n	Number with which to identify the attribute (1 through 20).
attr	Attribute to define. See Field Attribute Parameters under "Field and Attribute Commands" in this subsection.

Example

This example defines attribute number one as having protected input and a color of white on blue:

```
def,1,(pr,whi/blu)
```

Field (FLD) Command

Use the FLD command to generate a field on the screen.

Format

```
FLD[,r,c,rsiz,csiz,o,attr,end-attr,text]
```

After the statement executes the FLD command, the cursor appears on the last line of the field, two columns beyond the end of the field.

Following is a description of the subfields:

Field	Description
r,c	Cursor position of the upper left corner of the field. See Syntax Rules for Screen Commands under "Screen Commands" for rules on specifying screen coordinates.
rsiz	Vertical field size in rows (lines). Default = 1.
csiz	Horizontal field size in columns (characters). Default = 1.
o	Options. Border options: A Attributes. Outputs field attributes even if the field is being boxed. Use this option to produce a highlighted box around a field. Use only with the B option. B Box. Draws a box around the field if the terminal supports box drawing characters. The box is drawn inside the perimeter of the field reducing the effective size of the field by two rows and columns. See also the A option. F Frame. Frames the field with emphasis. This option is terminal dependent. S Sides. Draws lines on the left and right sides using emphasis. This option is terminal dependent. T Top. Draws a line at the top of the field using emphasis. This option is terminal dependent. U Underline. Draws a line under the field using emphasis. This option is terminal dependent. **BTOS:** The border options (F, S, T, U) are not supported.

continued

continued

Field	Description
	Justification options: C Centers the text in the field. L Left-justifies the text in the field. R Right-justifies the text in the field. V Varies the size of the field to fit the text supplied. Default (no text) and maximum size is *csiz*. The field is left-justified within the columns bounded by *c* and *csiz*; use this with the R or C options to alter justification. Miscellaneous options: E Empty. If the field is empty (no text), do not display it. Use this option for status fields that should only be displayed if relevant. P Protect. Two conditions apply: 1. If a field is not defined with the PR parameter and is empty when placed on the screen, protect it from user input. 2. If a field is defined with the PR parameter, allow data that is supplied on the SC run statement via the statement option T.
attr	Attribute that defines the characteristics of the field. (See Attribute Parameters.) If no attribute is specified, no attributes are output.
end-attr	Attribute that terminates the field. This parameter is not required.
text	Literal text to be inserted into the field. If you do not specify text, the data within the boundaries of the field remain unaltered. If you use the T option on the SC statement, text supplied in the ***fldtxt*** field of the SC statement is used instead. See also the P option.

Example

This example shows how the FLD command is used to create the Add Report (AR) function form. The A, B, and F options in the first FLD command create a highlighted border if permitted by the terminal type. The second FLD command places the menu title in the center of the top line of the border. Execute the AR function to see the result.

```
prep,(pr,bac)
fkey,1,Resume,rsm
fkey,2,Paint,pnt
fkey,4,Return,formret,7
FKEY,8,Help,dsphelp,1
fkey,10,Quit,^
PC,1,61;'AR'

FLD,2,6,6,69,afb,(PR,boc)
FLD,2,6,,69,cv,(PR,tc),,' Add Report '

dfld,inp,u,(ts,ai,fc);dfld,,,(co,fc)
area,,3,8,4,65,,(pr,mc)

 Report or drawer  __inp_________
            Title  __inp_____________________________________ -
end
```

(continued)

Message (MSG) Command

Use the MSG command to display a message or prompt line.

Format

MSG, ***r*** **,** ***o*** **,** ***attr*** **,** ***text***

After the statement executes the MSG command, the cursor appears in column 1 of the line on which the message is displayed.

Following is a description of the subfields:

Field	Description	
r	Row on which to place the message.	
o	Options:	
	Blank	Displays message text as given.
	B	Blink. Allows blink characters to be used on non-error messages. See the LB and RB commands in "Text Handling Commands."
	E	Presents the text in system message format. That is, center the message text on the line and translate the < and > characters to left and right blink characters.
	L	Left-justifies the message text.
	R	Right-justifies the message text.
	C	Centers the message text on the line.
attr	Attribute to be used on the message line. The default attribute is reverse video (RV) and the user's configured system message color from the user registration report. **1100:** Default = white on red. **BTOS:** Default = white on black.	
text	Literal text for the message (up to 79 characters).	

Example

This example displays a message on line 1 using the text supplied. The user's default message attribute is used.

```
msg,1,e,,'that input is not valid'
```

Prepare Screen for Painting (PREP) Command

The PREP command prepares the screen for painting an input menu. The PREP command clears the screen and outputs an attribute on row 2, column 1.

If you do not specify an attribute, (PR,WHI/BLA) is assumed. The PREP command is equivalent to this sequence of commands:

```
CS;ATT,(attr),2,1.
```

Format

```
PREP[,attr]
```

Following is a description of the subfield:

Field	Description
attr	The attribute to use in place of the default. See Field Attribute Parameters under "Field and Attribute Commands" in this subsection.

Example

This example protects the input menu being created by subsequent screen commands and displays the background colors as defined by the Terminal Definition Report for the terminal type.

1100: On OS 1100 MAPPER Systems, displays the background colors as defined by the terminal type.

```
prep,(pr,bac)
```

Define Field (DFLD) Command

The DFLD command defines field characteristics for later use by the AREA command. The DFLD command supplies all the information necessary to output a field, except screen location and field size which is supplied by the AREA command. After an AREA command is processed, all field definitions are discarded. See "DFLD and AREA Commands Example" in this section.

You can define up to 40 named and 40 unnamed fields using the DFLD command.

Format

```
DFLD[,name,opt,attr,text]
```

Following is a description of the subfields:

Field	Description
name	The name of the field you are defining. You can use a maximum of 12 characters.
opt	Options.
	Border options (terminal dependent):
	F Frame. Frames the field with emphasis.
	S Sides. Draws lines on the left and right sides using emphasis.
	T Top. Draws a line at the top of the field using emphasis.
	U Underline. Underlines the field using emphasis.
	Justification options:
	C Centers the data in the field.
	L Left-justifies the data in the field.
	R Right-justifies the data in the field.
	V Varies the size of the field to fit the text supplied. Default (no text) and maximum size is determined by the length of field as specified in the AREA command data. The field is left-justified (use with R or C options to alter justification).

continued

continued

Field	Description
	Miscellaneous options: E Empty. If the field is empty (no text), do not display it. P Protect. Two conditions apply: 1. If a field is not defined with the PR parameter and is empty when placed on the screen, protect it from user input. 2. If a field is defined with the PR parameter, allow data that is supplied on the SC run statement via the statement option T.
attr	Attribute that defines the characteristics of the field. (See Attribute Parameters.) If no attribute is specified, no attributes are output.
text	Literal text to be inserted into the field. If you use the T option on the SC statement, text supplied in the ***fldtxt*** field of the SC statement is used instead. See also the P option. If you do not supply text, the field will be blank.

Area (AREA) Command

Use the AREA command to lay out an entire region (area) of the screen including both its text and fields.

The data that defines the contents of the area begins in column 1 of the report line immediately following the AREA command. The text is displayed exactly as it appears. The locations and sizes of fields within the area are indicated by low line characters (_).

The AREA command recognizes reserved words, inline attributes, and literal delimiters in its data.

Note: *The AREA command is valid only if screen commands are being read from a report.*

Condition

The characteristics of the fields indicated with the AREA command must be previously defined with DFLD commands.

Format

```
AREA[ ,name ,r ,c ,rsiz ,csiz ,opt ,attr ]
```

Following is a description of the subfields:

Field	Description
name	The name of the area you are defining. You can use a maximum of 12 characters. The MAPPER system stores up to 10 named areas in memory. Naming an area alters the way input is received from the screen. When XKEY$ or a function key is pressed with the cursor in a named area, only the input from that portion of the screen is received. If the area is output from a run: - AREA$ contains the name of the area from which input occurs. - FIELD$ contains the field number within the area in which the cursor is located. - CURV$/CURH$ contains the cursor coordinates relative to the upper left corner of the area.
r,c	Cursor position coordinates for the upper left corner of the area. See Syntax Rules for Screen Commands under "Screen Commands" for specifying screen coordinates.
rsiz	Vertical area size in rows (lines). Default = 1.
csiz	Horizontal area size in columns (characters). Default = 1.
opt	Options. Border options: A Attributes. Displays the background attributes of the area even if the area is being boxed. Use only with the B option. B Box. Draws a box within the outer edge of the area if the terminal supports box drawing characters. See also the A option. F Frame. Frames the area with emphasis. This option is terminal dependent. U Underline. Underlines the area using emphasis. This option is terminal dependent. Miscellaneous options: D Delimiter. Marks fields with a field delimiter character only. Named fields are not recognized. Allows you to place text immediately following a field.
attr	Attribute that defines the background characteristics of the area.

How Field Definitions Work

If a named field is indicated in the area data, the characteristics of the field come from the DFLD command specifying the same field name. A named field definition may be referenced more than once.

The characteristics of unnamed fields in the area are supplied by DFLD commands that specify no field name. The first unnamed field indicated in the area uses the first unnamed definition; the second unnamed field uses the second unnamed definition, and so on. If there are more unnamed fields indicated than there are unnamed definitions, the last unnamed definition is used to satisfy the remaining fields.

After the AREA command is performed, all field definitions are discarded.

DFLD and AREA Commands Example

This example uses the DFLD and AREA commands to lay out an input menu. Note that each field begins with at least one low line character. When named field references are indicated in the area data, the name is included in the size of the field.

The first unnamed field (`Drawer letter`) uses the first unnamed definition; the second unnamed field (`User-id`) uses the second unnamed definition. The last menu field (the single-character field following the `Last Report` field) uses the third unnamed definition. The FLD commands put the border and title around the menu; this must be done before the AREA command.

```
fld,3,9,10,60,afb,(pr,rv,boc)
fld,3,9,,60,cv,(pr,tc),,' Index User '

dfld,num,u,(ts,no,fc)
dfld,date,u,(ts,ai,fc)
dfld,,u,(ts,ao,fc)
dfld,,u,(ts,ai,fc),user$
dfld,,,(co,fc)
area,,4,11,8,56,,(pr,mc)

    Number of lines from each report _num_
                        Drawer letter _
                  User-id (ALL for all) ___________
    Last update start date (DDMMMYY) _date__
      Last update end date (DDMMMYY) _date__
                         First report _num
                          Last report _num _
end
```

Field Attribute Parameters

Attributes control the type of input allowed in a field (alpha, numeric, cursor, protected), and they control intensity, color, text justification, tab settings. There are six categories of field attribute parameters.

- Input control
- Intensity
- Text input justification
- Tab stop
- Emphasis protection
- Color

Input Control Parameters

AI Any input allowed (default).
AO Alphabetic input only.
NO Numeric input only.
CO Cursor only — no input allowed. The user can move the cursor into the field, but cannot type any input. Use to create menu selection items.

1100: The CO parameter is not available on OS 1100 MAPPER Systems.

KO Kanji input only allowed. This parameter is only meaningful for terminals that support Kanji.

1100: The KO parameter is not available on OS 1100 MAPPER Systems.

PR Protected — cursor cannot be placed in field.

Intensity Parameters

BL Blinking.
LI Low intensity.
NI Normal intensity (default).
RV Reverse video.
VO Video Off. Use to make text invisible.

BTOS: BL, LI, NI, and RV parameters are not supported.

Text Input Justification Parameter

RJ Right-justify data entered by the user.

Tab Stop Parameter

TS Tab stop. Unlike a tab character, an attribute tab stop does not require a screen location, so the cursor stops at the attribute position rather than immediately after it.

Emphasis Protection Parameter

UE Unprotected emphasis. Overrides the normal protection given to emphasis. The UE parameter causes the emphasis to be destroyed if the data at that location is erased or typed over.

Color Parameters

Specify color parameters in pairs, separated by a slant (/). For example, WHI/RED specifies white characters on a red background. If you do not specify a color, white characters on a black background is assumed.

BLA	Black
BLU	Blue
CYA	Cyan
GRE	Green
MAG	Magenta
RED	Red
WHI	White
YEL	Yellow

BTOS: Seven colors, plus black, are supported in the text mode. The only available background color is black. If you select black as the character color, the selected background color is used instead against a black background. If you select the same color for characters and background, green characters on a black background are used instead.

Terminal Defined Color Parameters

The following parameters determine the foreground and background colors of menus, reports, fields, or field text defined by the terminal type in the Terminal Definition Report or in the User Registration Report. These parameters free the screen designer from having to choose the color for each element on the screens.

1100: On OS 1100 MAPPER Systems, these parameters determine the foreground and background colors as defined by the terminal type.

BAC	Background color.
BOC	Border color.
TC	Title color.
MC	Menu color. (The background color inside the menu border.)
FC	Field color. Use this parameter with the input editing parameters (AI, AO, NO, CO, KO, and PR) to select a color.
RC	Report color. This parameter selects either the user's configured report or result color based on the current report (-0) of the station.

1100: Selects the color based on the current report (-0).

Using Inline Attributes

Use inline attributes in freeform text to highlight words or phrases. Create inline attributes using the tilde (~) character followed by any valid field attribute parameter. See Field Attribute Parameters under "Field and Attribute Commands" in this subsection for a complete list of parameters.

Inline attributes apply only to the DATA and AREA commands.

Example

In this example, the freeform text appears as reverse video and yellow on black. When the text is displayed, it starts at the cursor position of the previous tilde.

```
~(rv,yel/bla)SC uses inline attributes.~(whi/bla)
```

Text Handling Commands

Table 7-16 lists the text handling commands that display literal text on the screen. You can also use them to substitute characters.

Table 7-16. Special Commands

Command	Description
SOE[,c | ,*n*]	Places a start-of-entry (SOE) character at the current cursor position. The value supplied for *n* is the number of SOEs to display. Default = 1. The value supplied for c is the special character that you are using to redefine the SOE character. The following example shows & as the special character to represent the SOE character. For example: soe,&;'Enter new value &'
LB[,c | ,*n*]	Places a left blink character at the current cursor position. The value supplied for *n* is the number of left blinks to display. Default = 1. The value supplied for c is the special character you are using to represent the left blink character. See the RB command for an example.
RB[,c | ,*n*]	Places *n* right blink characters at the current cursor position. Default = 1. The value supplied for c is the special character you are using to represent the right blink. For example: lb,*; rb,%; MSG,1,e,,'* Valid operators are <,>,= %' produces the following: < Valid operators are <,>,= >
TIC[,c | ,*n*]	Places *n* current literal delimiter characters at the current cursor position. Default = 1. Redefines the literal delimiter character, which is initially the apostrophe, to a special character supplied by c. This is useful when apostrophes are used in text. For example: @sc '' 'tic,"; "It is a sunny day."' tic,"; "It's a sunny day."

EMP Command

Use the emphasis command to highlight or emphasize text on the screen. You can place emphasis anywhere on the screen.

Format

```
EMP[,p,p,...,p]
```

Following is a description of the subfield:

Field	Description	
p	Emphasis parameter or parameters, as follows:	
	Blank	Turns off emphasis
	C or |	Separates columns
	U or _	Underscores
	-	Strikes through

BTOS: Hardware display intensity is not supported.

Example

This example underscores the text enclosed in the literal delimiters.

```
emp,_;  'This text is underscored'
emp; 'This text has no emphasis'
```

DATA Command

Use the DATA command to display a specified number of lines of text data starting at a specified screen position.

The text data to display begins in column 1 of the report line immediately following the DATA command. The text is displayed exactly as it appears; spaces or commas within the data do not terminate the scan.

Note: The DATA command is valid only if screen commands are being read from a report.

Format

```
DATA[,r,c,rsiz,csiz,pn,o]
```

Following is a description of the subfields:

Field	Description
r,c	Cursor position at which to start displaying the text. See Syntax Rules for Screen Commands under "Screen Commands" for rules on specifying screen coordinates.
rsiz	Number of rows on the screen for the displayed text (paged text if the *pn* parameter is specified). Default = 1.
csiz	Number of columns on the screen for the displayed text (paged text if the *pn* parameter is specified). Default = report line length, or screen width, whichever is smaller.
pn	Page number. Indicates that the following report lines consist of page-formatted data, and specifies the default initial page to be processed. This page number may be overridden. See DSPFORM Action and PAGE Action in this subsection. Use paged data to handle multiple menus within the same report, or for textual information too lengthy for a single screen, such as application documentation.
o	W option. Interpret reserved words, inline attributes, and literal delimiters in DATA text.

#PAGE Identifier

A page identifier that marks the beginning and end of a form page.

Format

```
#PAGE[,n]
```

Following is a description of the subfield:

Field	Description
n	Optional page number.

Comments

- The identifier must begin in column 1 of the report and the word PAGE must be in uppercase (#page does not work).
- Pages may be numbered or unnumbered. However, only a numbered page can be used as the initial page. Once the initial page is displayed, the PAGE action allows access to any page.

Example

Page data can be structured as follows:

SC format	Section description
#PAGE,1	Page identifier
Screen commands	Pretext section
END	End of pretext section
Number of lines specified by *rslz* on DATA command	Page text section (optional)
Screen commands	Post-text section (optional)
#PAGE,2	Page identifier

How the SC Statement Interprets Paged Data

- Processing of paged data always stops when the next page identifier is encountered.
- If the DATA command specifies a row size (*rs* / *z*) or column size (*cs* / *z*) of zero, the pages are assumed to consist only of a single command section.
- Run statement options in effect at the start of the initial page remain in effect with each new page. See the FKEY action PAGE.
- Begin context-sensitive help (help for a form field) for a specific page after an END command and before the next page identifier.
- When moving from one page to the next,
  - Current function key mapping is retained.
  - Individual function keys can be mapped or remapped.

Paged DATA Example

In this example, the DATA command displays page 1 unless a DSPFORM action specifies a different page number. The *pn* field in the DATA command contains the value for the default page.

```
data,,,,,1
#PAGE,1
fld,1,1,,65,,,,' MAPPER Online Documentation'
fld,1,66,,14,,,,'Screen 1 of 1'

 (more screen commands)


#PAGE,END
```

Screen Printing Commands

This section on screen printing commands does not apply to the BTOS II MAPPER System or the Personal Computer MAPPER System.

The following screen printing commands control output to a printer:

PRT Sends protected and unprotected data to the attached print device. It sends all data from the home position up to and including the character under the cursor. The station number of the device must exist in the MAPPER configuration report.

PRF Sends unprotected data to the attached print device (such as COP). The PRF command is the same as the PRT command, but it sends only unprotected data.

LF[*n*] Places *n* line feed characters at the current cursor position. Default = 1. Use this command with the PRT and PRF screen commands.

FF Places a form feed character at the current cursor position. Use this command with the PRT and PRF screen commands.

Setup Commands

The setup commands perform a variety of tasks such as sounding a beep at the terminal, altering terminal setup, stopping the scan for screen commands, controlling graphic display modes, overriding current options, and mapping function keys.

BEEP Command

The BEEP command causes the terminal beeper to sound once. Use it to signal the terminal user that an error or other special condition has occurred.

Format

```
BEEP
```

After your run executes the BEEP command, it places the cursor at the home position. You can execute only one BEEP command for each SC statement.

Note: *Not all terminals support this functionality.*

CP Command

This section on the CP command does not apply to the BTOS II MAPPER System or the Personal Computer MAPPER System.

The CP command alters the terminal setup (control page).

Format

```
CP[,p]
```

The CP command accesses the terminal setup and inserts the text string *p*. Any special sequence (control string introducer) required is inserted automatically.

Note: *Be careful when altering the terminal setup; it may affect the ability of the MAPPER system to communicate with the terminal. See your terminal documentation before using this command.*

END Command

The END command stops the scan for screen commands and displays the terminal output.

Format

```
END
```

MODE Command

The MODE command selects the display mode of a graphics terminal. (Nongraphics terminals ignore the MODE command.)

Format

```
MODE[,p]
```

The valid entries for the *p* field are as follows:

A	Alpha only display.
G	Graphics only display.
M	Mixed mode display (graphics and alpha). This mode is terminal dependent.

OPTS Command

The OPTS command overrides a subset of SC statement options that may be in effect from the calling run. It allows the screen command report to specify a subset of the screen control statement options independent of the calling run.

Format

```
OPTS,[+ | -]o
```

If the first option character is a plus (+) the specified options are added to the caller's options; a minus (-) removes the specified options. Allowable options are B, C, H, K, L, M, S, U. See Options in this subsection for a description of each allowable option.

Example

This example holds line control and displays all blank spaces. All other options are deselected.

```
opts,ls
```

FKEY Command

The FKEY command specifies an action to be performed at the next input if a particular function key is pressed. This is called function key mapping. You can map actions to function keys 1 through 10 as well as **Transmit**. If one or more keys are mapped, the remaining unmapped function keys are disabled.

The FKEY command also displays a function key bar at the bottom of the screen that shows the current mapping of function keys 1 through 10. The current mapping remains in effect until the next output (see the K option under Options).

Format

FKEY,*n*,*title*,*action*

Following is a description of the fields:

Field	Description
n	Function key number (0 = **Transmit**).
title	Text to display with the function key number on the function key bar. The title must be six characters or less. Enclose it in apostrophes. For example: **fkey,8,'Ex hlp',^**
action	The action to take when the function key is pressed. The action can be any valid command (control line) input that a user might enter, such as a ^ to display the active screen or an x to sign off MAPPER software. See "Special FKEY Actions" in this subsection for other possible actions.

Special FKEY Actions

The possible actions that may be specified with the FKEY command are as follows:

- Display Help (DSPHELP)
- Display Form (DSPFORM)
- Return to Previous Form (FORMRET)
- Drawer Select (DRWSEL)
- PAGE
- SELECT
- KEY

DSPHELP Action

Use the DSPHELP action to display context-sensitive help text (help for a form field). The context-sensitive help text must be in the same report as the FKEY command using the DSPHELP action and is preceded by the HELP identifier.

Format

```
DSPHELP
```

Example

This example displays context-sensitive help for the field where the cursor is positioned when the **F8** key is pressed.

```
fkey,8,Help,dsphelp
```

Note: See "HELP Identifier."

HELP Identifier

Use the HELP identifier to identify help text in your screen command report and to determine where the context-sensitive help should be placed on the screen.

Format

```
HELP,fld,...fld row
```

Following is a description of the fields:

Field	Description
fld,...fld	The field number for which the following context-sensitive help refers.
row	The row (screen line) where the help text display should start.

Example

In this example, the END command separates the commands used to build the screen from the help text. The DSPFORM action specifies another form to display if the user presses **Help** while the context-sensitive help (help for a form field) is already displayed. The | character on the first HELP identifier determines the width of the help window.

```
END
DSPFORM,49H,1
HELP,1 15                                    |
                First Field

This is the text for the first field on
the screen.  It is freeform text.
HELP,2
              Second Field

This is the text for the second field on
the screen.
HELP
```

DSPFORM Action

Use the DSPFORM action to access a report containing screen commands (also referred to as a form) when a function key is pressed.

While using the DSPFORM action, a stack of forms, built up in the order in which they are displayed, is created. The stack grows this way until it is cleared or rearranged using the DSPFORM or FORMRET FKEY actions. The form at the top of the stack is the currently displayed form.

This stack is called the forms return stack and identifies which form MAPPER software should return to if a FORMRET action is used. A stack holds a maximum of 20 forms.

A form on the stack can include a saved context so that it can be returned to the top of the stack or recalled at any time while a function is still active. The forms that were stacked on top of it are discarded.

Format

DSPFORM, ***r*** **[** ***dc*** **],** ***pn*** **,** ***tabp*** **,** ***opt***

Following is a description of the fields:

Field	Description
r[dc]	The report, drawer, and cabinet containing the form. If the drawer or cabinet or both are omitted, DSPFORM action assumes the cabinet number and the drawer letter of the report containing the DSPFORM action. (This allows forms to be created for maximum portability.) The system begins reading commands on line 4 of the report you specify (be sure to take this into account when entering commands in the output area).
pn	The form page number of text to display. This is valid only if the report specified by *rdc* uses paged data. See "DATA Command" in this subsection.
tabp	The tab position at which to place the cursor, where a positive number is the number of tab positions forward (maximum of 100) from the home position and a negative number is the number of tab positions backward (maximum of 100) from the home position. Default = 1. 0 = no positioning.

continued

continued

Field	Description
opt	An option to set the return to this form if requested by a later form. Use one of these values to specify the action for the return stack: 0 Puts this form on the top of the stack. 1 Clears the stack and puts this form on the bottom. 2 Marks the current stack position and then puts this form on the top of the stack. 3 Saves the context of the current form so a return restores the screen as it currently exists. This is useful for help displays. **1100:** This option is not available on OS 1100 MAPPER Systems. 4 Does not put an entry on the return stack for this form. 5 Overwrites the current entry on the return stack with this form.

Example

This example allows the user to display page 4 of the form in 202E by pressing the **Readme** key (the **F6** function key).

```
fkey,6,Readme,dspform,202e,4
```

Displaying a Form and Returning Control to the Run

This section on displaying a form and returning control to the run applies only to the OS 1100 MAPPER System.

Use the Display Form (DSF) statement to display a report containing screen commands (form) from within a run. Use the DSF statement instead of the SC statement to display a form starting at a particular page number.

Format

```
@DSF,c,d,r,pn,tabp,opt .
```

The subfields for the DSF statement are the same as the DSPFORM action, with the exception of the `c,d,r` subfields (cabinet, drawer, and report containing the form).

FORMRET Action

Use the FORMRET action to return to a previous form as determined by the forms return stack. See "DSPFORM Action" in this subsection for more information on stacking.

Format

```
FORMRET, cmd
```

Following is a description of the field:

Field	Description
cmd	A value that acts on the display. The values are
	0 Returns to the previous entry on the return stack.
	1 Returns to the start of the stack and clears the stack.
	2 Returns to a mark previously set by a DSPFORM action with an *opt* value of 2 (see "DSPFORM Action" in this subsection).
	3 Returns to a saved context set by a DSPFORM action with an *opt* value of 3 (see "DSPFORM Action" in this subsection). Note that only one saved context applies for a user at any time.
	1100: This value is not available on OS 1100 MAPPER Systems.
	4 Returns to the top entry on the stack. This is used if the current command does not put anything on the stack.
	5 If there is a report on display, repaints the report before redisplaying the form. This is used to clean up a screen that has several menus on it.
	6 If there is a report on display, either repaints it, or does a formret,0.
	7 If there is a report on display, either repaints it, or does a formret,4.

Example

In this example, pressing the **Return** function key returns the user to the previous entry on the return stack.

```
fkey,4,Return,formret,0
```

PAGE Action

Use the PAGE action to page forward or backward in a form, after the initial page of a form is displayed.

Format

```
PAGE,p
```

Following is a description of the field:

Field	Description
p	Use a plus (+) to move forward to the next page, a minus (-) to indicate the previous page, or a number to go to a particular page. Default = +.

Example

In this example, pressing the **RollFw** key advances the user to the next page of paged data.

```
fkey,2,RollFw,page,+
```

SELECT Action

Use the SELECT action to select one of a group of actions or commands, based on the relative position number of the field that the cursor is in at the time the key is pressed.

Format

```
SELECT
```

Note: *The SELECT command must be in uppercase letters and the field number must not start in column one.*

Example

This example displays form 11e if the cursor is in the first field when the **Transmit** key is pressed, and sorts the report on display if the cursor is in the second field when the **Transmit** key is pressed.

```
fkey,0,,SELECT
  1,dspform,11e,1,1,0
  2,sort -
```

1100: The following example applies only to OS 1100 MAPPER Systems.

This example displays form 11e if the cursor is in the first field when the **Transmit** key is pressed, and returns control to the run if the cursor is in the second field when the **Transmit** key is pressed. FKEY$ contains zero and FIELD$ contains 2.

```
fkey,0,,SELECT
  1,dspform,11e,1,1,0
  2,KEY
```

KEY Action

Use the KEY action to allow the function key to be passed directly to a run. This action allows function keys to be interpreted by the run instead of by the controlling form.

Format

```
KEY
```

Notes: *1. The KEY action must be in uppercase letters.*

2. The FKEY$ reserved word can be used by the run to determine which key was pressed.

Example

This example passes the **F1** key to the run.

```
fkey,1,Cont.,KEY
```

SCH (Schedule)

This section does not apply to the OS 1100 MAPPER System or the BTOS II MAPPER System.

The Schedule Runs (SCH) statement allows you to establish a time for the execution of a specified run statement. The SCH statement can queue any of the following run statements for execution at a later time:

- Auxiliary (AUX)
- Background Run (BR)
- Print (PRT)
- Remote Run (RRN)
- Send Report (SEN)
- Send Report to User (SNU)
- Start (STR)

Format

```
@SCH[,date,time] rst .
```

Following is a description of the field and subfields:

Field	Description
date	Date to schedule the queuing function in YYMMDD (DATE1$) format. Default = current date.
time	Specifies the time of day to schedule the queuing function in HHMMSS format where HH is the hour, MM is the minute, and SS is the second. Default = current time.
rst	Run statement to queue: AUX, BR, PRT, RRN, SEN, SNU, STR.

Example 1: Schedule a Background Run

Schedule the background run named status, with input data apr and monthly, for execution at 7:00 p.m. on June 3, 1990:

```
@sch,900603,190000 br status,apr,monthly .
```

Example 2: Schedule a Print

Schedule the current -0 result to be printed on the current date at 10:00 p.m.:

```
@sch,,220000 prt,-0 .
```

SEN (Send Report)

The Send Report (SEN) statement sends an entire report as a message to a station. You can send reports only to configured stations.

If you send a report to a station within your site, the SEN statement creates a result in the same drawer as the original report. If you send a report to a remote site, the statement creates a result in cabinet 0, drawer F.

1100: On OS 1100 MAPPER Systems, you cannot send messages to a remote site.

Format

```
@SEN,c,d,r,sn[,sl,ack?,lab] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report to send.
sn	Station number to which to send the report.
sl	Site letter. Default = local site. **1100:** This subfield is not available on OS 1100 MAPPER Systems.
ack?	Return acknowledgement when the message is received, Y or N. Default = N.
lab	Label to go to if the station is not configured.

Examples

Send report 1A0, to station number 5:

```
@sen,0,a,1,5 .
```

Send result -2 to station number 22 and return an acknowledgement after the message is accepted (go to label 99 if the station does not exist):

```
@sen,-2,22,,y,099 .
```

1100: The following example does not apply to OS 1100 MAPPER Systems.

Send report 2C0, to station number 5 at MAPPER site j (go to label 3 if the station does not exist):

```
@sen,0,c,2,5,j,,003 .
```

SFC (Set Format Characters)

The Set Format Characters (SFC) statement establishes the format for a following output display, such as that created by a Display Report (DSP) or Output Mask (OUM) run statement.

1100: On OS 1100 MAPPER Systems, you can also select columns to print the next time you call the Auxiliary (AUX) or Print (PRT) run statements.

Format

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

Following is a description of the field and subfields:

Field	Description
c,d,r	Report on which to set the format. Default = -0.
vld	The columns of the report you want to include in the display format. Use variables, literal data, or both to specify the format you want to set.

Comments

- You can restore a display format previously saved with a Load Format Characters (LFC) statement or build your own customized format with an SFC statement. The **x** characters (or any characters other than spaces) become displayed column positions; spaces are column positions not displayed.
- You must use a DSP, DSX, OUM, or OUT statement after an SFC statement.
- If a report is specified on the SFC statement, that report becomes the current -0.

Example

Set the format for columns 1 through 10 and 16 through 20 in the current result and display it:

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

SNU (Send Report to User)

The Send Report to User (SNU) statement sends an entire report or result as a message to another user.

If you send a report to a station within your site, the SEN statement creates a result in the same drawer as the original report. If you send a report to a remote site, the statement creates a result in cabinet 0, drawer F.

1100: On OS 1100 MAPPER Systems, you cannot send messages to a remote site.

Format

```
@SNU,c,d,r,user[,dept,sl,ack?,lab] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report to send.
user	User-id to send the report to.
dept	Department number of the user. Default = all departments.
sl	Site letter. Default = local site. 1100: This subfield is not available on OS 1100 MAPPER Systems.
ack?	Acknowledgment requested after the message is received, Y or N. Default = N.
lab	Label to go to if the user-id or department does not exist.

Comments

- The label is taken only if the user-id or department does not exist on the local site. For remote sites, a message is returned informing you that the user-id or department is invalid.

1100: On OS 1100 MAPPER Systems, you cannot send messages to a remote site.

- If you do not specify a department number, the message is sent to that user-id in all departments. The first user to accept and acknowledge the message clears the message for all users.

Examples

Send report 1C0, to all users with the user-id of newuser (go to label 3 if newuser is not a valid user-id):

```
@snu,0,c,1,newuser,,,,003 .
```

1100: The following example does not apply to OS 1100 MAPPER Systems.

Send report 2B0, to mapcoord user-id in department 2 at site z, requesting an acknowledging response:

```
@snu,0,b,2,mapcoord,2,z,y .
```

SOR (Sort)

The Sort (SOR) statement sorts report or result data and creates a result.

1100: On OS 1100 MAPPER Systems, the system places an update lock on the report to prevent other users from updating it while your run is sorting it.

You can use the Sort and Replace Report (SRR) statement to replace the sorted data into the original report. You cannot use the SRR statement on results, and the SRR statement does not create a result.

Format

```
@SOR,c,d,r o cc ltyp,p .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report to sort.
o	Options field (see Options).
cc	Column-character positions or names of the fields to sort.
ltyp	Line type to sort (if you specify the A option, you can leave this subfield blank, but enter the comma).
p	Sort parameters (1 through 5, N, D). Enter sort parameters (1-5) in ascending order starting with 1. You can also use the N and D parameters: N = numeric sort D = descending order

Options

A	Processes all line types.
C(S)	Distinguishes between uppercase and lowercase letters.

1100: C(x) Alters the sort process based on the character set order. Ordinarily the system processes the report based on the character set of the drawer. The C option allows you to choose the character set type on which to base the sort. Use one of the following:

C(F)	Full character set (FCS)
C(L)	Limited character set (LCS)
C(S)	Strict comparison; distinguishes between uppercase and lowercase letters.

1100: X+ Sorts the data using the OS 1100 Sort/Merge product (if installed at your site), regardless of the report length.

1100: X- Sorts the data using the normal MAPPER software process, regardless of the report length.

If you do not specify X+ or X-, the sort process is determined by the report length. See the coordinator for more information.

Example 1: Sort by Field in Ascending Order

Sort a report in ascending order by the Cust Code field:

```
@sor,0,b,2 '' 'cust' □,1 .
```

where:

`''`	Uses no options
`'cust'`	Sorts the Cust Code field (column 45 for four characters)
`□`	Processes tab lines only
`1`	Sorts one level in ascending order

Example 2: Sort a Report by Two Fields

Sort a report first by the Cust Code field, then by the Product Type field:

```
@sor,0,b,2 '' 'cust','product' □,1,2 .
```

or

```
@sor,0,b,2 '' 45-4,15-9 □,1,2 .
```

where:

`''`	Uses no options

`'cust'`
`45-4`
`1`
	Sorts the Cust Code field (column 45 for four characters) first

`'product'`
`15-9`
`2`
	Sorts the Product Type field (column 15 for nine characters) second (that is, within the Cust Code field)

`□`	Processes tab lines only

Using the ISOR Run to Create an SOR Statement

This section on using the ISOR run applies only to the OS 1100 MAPPER System.

You can use the Iterative Sort (ISOR) run to create an SOR statement. For information on how to use the iterative runs, see the *Manual Functions Reference*.

SRH (Search)

The Search (SRH) statement searches vertically through one or more reports or one result and creates a result of items found.

To create an update result that allows you to blend the changed lines back into the original report or delete lines from the original report, use the Search Update (SRU) statement. The SRU statement uses the same fields and subfields as the SRH statement. See the online help system (HELP,@SRU) for more information.

Format

@SRH,*c*,*d*[,*r*,*l*,*q*,*lab*] *o cc ltyp*,*p* [*vlines*,*vls*,*vrpt*] .

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report to scan.
l	Line number at which to begin the scan.
q	Number of lines to scan.
lab	Label to go to if no data is found.
o	Options field (see Options).
cc	Column-character positions or names of the fields to scan.
ltyp	Line type to scan (if you specify the A option, you can leave this subfield blank, but enter the comma).
p	Search parameters.
vlines	Variable to capture the number of lines found with the search parameters.
vls	Variable to capture the number of lines that were scanned. If a start scan line is specified, all lines scanned (not just line type specified) are included.
vrpt	Variable to capture the report number where the find was made. The ***vrpt*** variable is useful only with the B[(*n*)] option to capture the number of the report containing the last find made before bailing out. Note that if you do not use the B option, ***vrpt*** = 0.

Options

A	Processes all line types. Do not use this option with the T or U options.
B[(*n*)]	Stops a search after the *n*th find. Default = first find. The search continues until *n* items are found or until the end of the report.
C(S)	Distinguishes between uppercase and lowercase letters.
1100: C(*x*)	Alters the search process based on the character set order. Ordinarily the system processes the report based on the character set of the drawer. The C option allows you to choose the character set type on which to base the search. Use one of the following: C(F) Full character set (FCS) C(L) Limited character set (LCS) C(S) Strict comparison; distinguishes between uppercase and lowercase letters.
D	Omits search information lines from the result.
1100: E*n*	Estimates the number of lines in the result, where *n* is an integer. This option improves the efficiency of the search when the output result contains more than 500 lines.
F	Searches for a numeric value instead of a character string. To search for a range of negative or floating point numbers, use this option.
H	In a multiple report search, includes the heading lines of the first report only.
L(*x*)	Omits lines of type *x* from the result. Note that you can use any line type here. For example, L(*.a) indicates that asterisk lines, period lines, and A-type lines are to be omitted from the result.

N — Includes only those lines not meeting search criteria. For example, if you search the column for the letter A, the result contains all lines that do *not* contain the letter A in the column. Do not use this option with the T or U options.

P — Extracts paragraphs from the report. A paragraph consists of data on a specified line type and all subsequent lines until the next occurrence of that line type.

Q[(*n*)] — Extracts the first *n* paragraphs from the report; stops the search after the *n*th paragraph. Default = first paragraph.

R*x*{-*y* | ','*y*} — Scans a range of reports from report *x* through report *y*; scans reports *x*,*y*,...*y*. (Note that the format R*x*,*y* requires apostrophes around the comma.)

1100: On OS 1100 MAPPER Systems, you do not need to enclose the comma in apostrophes.

Examples:

r2','5	Scan reports 2 and 5
r2-10	Scan reports 2 through 10
r2-10','14	Scan reports 2 through 10 and 14

1100: S — Groups lines found in search parameter order. (Use the S option with the H option to avoid repeating report headings for each group.) This option does not apply to the SRU statement.

T[(*x*)] — Includes data found on the line type you are processing and on the preceding *x* type line. Default = tab line. Do not use this option with the A, N, or U options.

A common use for the T option is to include the tab line preceding the asterisk line found.

U[(*x*)]	Includes data found in the type of line you are processing and its surrounding data unit. A data unit consists of a group of lines starting at line type *x* up to the next line of that type. Default = tab line. Do not use this option with the A, N, or T options. With the U option, you specify the paragraph boundary as the *x* line type. In contrast, the P option automatically uses the line type you are processing as the paragraph boundary.
@	Searches for spaces or a specific line type when used with the @ parameter.
/	Searches for a slant character as data when used with the / parameter.

Example 1: Searching a Field for an Item

Search for amco in the Cust Code field:

```
@srh,0,b,2 d 'cust' □,amco .
```

where:

`d`	Uses the D option to delete search information lines
`'cust'` `amco`	Searches the Cust Code field for the character string amco
□	Processes tab lines only

Example 2: Searching a Field for Items within a Range

Search the Order Number field for all numbers in the range 80000 to 90000:

```
@srh,0,b,2 d 39-5 □,80000/r,90000 .
```

where:

`d`	Uses the D option to delete search information lines
`39-5`	Searches the Order Number field (column 39 for five characters)
□	Processes tab lines only
`80000/` `r,90000`	for all numbers in the range 80000 to 90000

Example 3: Searching a Field for Two Items

Search for status codes sh or sc in the St Cd field and capture the number of lines found and the number of lines scanned:

```
@srh,0,b,2 d 2-2 □,sh/□,sc <found>i3,<scanned>i3 .
```

where:

`d`	Uses the D option to delete search information lines
`2-2`	Searches the St Cd field (column 2 for two characters)
`□,sh/□,sc`	for sh or sc on tab lines
`<found>i3`	Captures the number of lines found in <found>
`<scanned>i3`	Captures the number of tab type lines scanned in <scanned>

Using the ISRH Run to Create an SRH Statement

This section on using the ISRH run applies only to the OS 1100 MAPPER System.

You can use the Iterative Search (ISRH) run to create an SRH statement. For information on how to use the iterative runs, see the *Manual Functions Reference*.

STN (Station Information)

The Station Information (STN) statement provides information about a specific station.

Format

```
@STN[,sn,lab vuser,vdepn,vdept,vttyp,vvsiz,vhsiz,vaspect,
vcolor,vgraph] .
```

Following is a description of the subfields:

Field	Description
sn	Station number or terminal type you want information about.
lab	Label to go to if the station does not exist.
*vuser**	Variable to capture the user-id of the station.
*vdepn**	Variable to capture the department name of the user.
*vdept**	Variable to capture the department number of the user.
vttyp	Variable to capture the terminal type name of the station. (These names are identical to those provided by the TTYPE$ reserved word.)
vvsiz	Variable to capture the vertical screen size of the station.
vhsiz	Variable to capture the horizontal screen size of the station.
vaspect	Variable (F type) to capture the aspect ratio of the specified station. (Same as the value of the ASPECT$ reserved word.)
vcolor	Variable to capture the graphics color flag or terminal. (Same as the value of the COLOR$ reserved word.)
vgraph	Variable to capture the graphics type. (Same as the value of the GRAPH$ reserved word.)

* If a user is not currently signed on to the terminal, *vuser* and *vdepn* are filled with spaces, and *vdept* is set to zero.

Comments

- If the station number you specify does not exist or is inactive, the run goes to the label specified in the *lab* subfield; otherwise, the run errs. If you do not specify a station number, or if the station number is zero, the STN statement assumes the number of the station where the run was started.
- If you specify a terminal type in the *sn* subfield, *vuser* and *vdepn* are blank; *vdept* contains 0.
- Note that the STN statement does not create a result. Any result that exists before the STN statement is executed is still the current (-0) result.

Example

Capture information about station 100:

```
@stn,100,199 <user>h11,<dname>h12,<dnum>i4,<term>h7,\
<vsize>i2,<hsize>i3 .
```

where:

100	Provides information about station 100
199	Goes to label 199 if station 100 does not exist.
<user>h11	Captures the user-id of the person currently signed on in <user>
<dname>h12	Captures the department name in <dname>
<dnum>i4	Captures the department number in <dnum>
<term>h7	Captures the terminal type (for example, PCHR) in <term>
<vsize>i2	Captures the vertical screen size (for example, 24) in <vsize>
<hsize>i3	Captures the horizontal screen size (for example, 80) in <hsize>

STR (Start)

This section does not apply to the BTOS II MAPPER System.

The Start (STR) statement collects data and uses this data as input to a job or processes this data as a job itself. It can also start jobs that produce files that can be returned to the MAPPER system with the Retrieve File (RET) statement.

PC MAPPER: The STR statement executes the command shell (CMD.EXE) contained in a MAPPER report.

Format

```
@STR,c,d,r[,sl,ifc,logon,psw] .
```

1100:

```
@STR,c,d,r[,runid,acct] .
```

Following is a description of the subfields:

Field	Description
c,d,r	Report that contains the job or the report to pass to a job.
sl	Site letter. Default = local site.
ifc	Interface name of the job to which to pass the report.
	A Series: Default = blank (execute the data as a WFL job). See the MAPPER system coordinator for the interfaces configured on your system.
	U Series and UNIX: Default = blank (**/bin/sh** — UNIX shell).
logon	Valid system logon to use at a remote site.
psw	System logon password. Required if the system logon has a password assigned.
1100: *runid*	Batch run identifier. Default = the run-id in the Exec RUN statement.
1100: *acct*	Batch run account number. Default = the account number in the Exec RUN statement.

Comments

- **U Series and UNIX:** When starting a job at a remote site, you must specify a valid system logon and password for the remote site.
- IBM® start requirements follow the examples.
- **A Series:** Some files generated by a WFL job can be returned to MAPPER software with the RET statement. You may also use the MAPPER sign-on command `RUN OBJECT/MAPPER("-f i l e n a m e")` using the name of a file containing a BPRUN$ statement to pass that file to MAPPER software for execution. If an interface name is used, a file title containing the data is given as a STRING parameter to the WFL job.
- If starting a batch job, the job must have been previously entered in a report or result.
- See the *Manual Functions Reference* and the online help system (HELP,START) for information on the BPRUN$ and ENDRD$ commands.
- To include more than one report or drawer, use the $INCL$ command. See Appendix C for information on data control commands.
- To control the format of the data in reports you transfer to the host, use the data control commands described in Appendix C.

Example 1: Starting a Job on the Local Site

1100: Start the runstream in report 101A0 and identify the batch run as bch10:

```
@s t r , 0 , a , 101 , bch10  .
```

U Series and UNIX: Start the batch runstream located in report 3B0 (the UNIX shell [/bin/sh] processes the runstream):

```
@s t r , 0 , b , 3  .
```

A Series: Start the WFL job located in report 3B0:

```
@s t r , 0 , b , 3 , , NEWUSER , LOCAL  .
```

IBM is a registered trademark of International Business Machines Corporation.

Example 2: Starting a Job on a Remote Site

U Series and UNIX: Pass report 101A0, to the job named filer on remote MAPPER site w. Use the logon of newuser with a password of remote:

```
@str,0,a,101,w,filer,newuser,remote .
```

A Series: Pass report 101A0, to the job named filer on remote MAPPER site w. Use the logon of NEWUSER with a password of REMOTE:

```
@str,0,a,101,w,filer,NEWUSER,REMOTE .
```

IBM Start Requirements

This section on IBM start requirements applies only to the U Series MAPPER System and UNIX MAPPER Systems.

If you intend to return output from the execution of the IBM jobstream, you must include an IBM ROUTE statement in the jobstream. See the UNIX system administrator for the remote address of the BSC or SNA connection.

If there is an error in the execution of the jobstream (that is, if BPRUN$ through ENDRD$ does not get built), the output from the IBM host is sent (through the BPERRIBM run) to the user MAPCOORD on the default site.

SUB (Subtotal)

The Subtotal (SUB) statement subtotals report data.

Format

```
@SUB,c,d,r[,lab] o cc ltyp,p vrslts .
```

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report in which data is to be subtotaled.
lab	Label to go to if no lines exist to be subtotaled.
o	Options field (see Options).
cc	Column-character positions or names of the fields to subtotal.
ltyp	Line type to process (if you specify the A option, you can leave this subfield blank, but enter the comma).
p	Subtotal parameters: + Add - Subtract * Multiply / Divide = Move to this field A Average C Cumulate M Move this field S Subtotal (required) S1-S5 Hierarchical Subtotals
vrslts	Variables to capture the results. Select the E option if you want the first variable to capture the entry count for each subgroup. Substrings are not allowed. To initialize a variable to the size of a field, just specify the variable number and type (for example, v1h, v2s, and so on). If you are using the E option, the variables must be fully defined (for example, v2i6).

Options

A Processes all line types.

C Looks for case changes in the subtotal key field values. Whenever the value in the subtotal key field changes, a subtotal is displayed. Use with the S parameter only.

> **1100:** On OS 1100 MAPPER Systems, this option applies only to full character set (FCS) reports.

E Counts the number of entries processed for each subtotal and enters that count in the first defined variable. For example, if a subtotal includes four values, the statement loads the first variable with 4.

I Ignores headings: processes report even though asterisk lines or heading divider line may not exist.

J(x) Justifies numeric result values in variables to x; x is one of the values listed below. Leading zeros appear to the left of a value, nonsignificant zeros to the right. Neither affect the value.

- C Eliminates leading and nonsignificant zeros. Inserts commas every three digits in the integer portion of the result. Right-justifies values.
- L Eliminates leading and nonsignificant zeros. Left-justifies values.
- R Eliminates leading and nonsignificant zeros. Right-justifies values.
- X Eliminates leading zeros, left-justifies values, then fills variable with nonsignificant zeros.
- Z Eliminates nonsignificant zeros, right-justifies values, and fills variable with leading zeros.

Comments

- Follow a SUB statement with an output line, since it creates one line of output for each item subtotaled. Do not use variable substrings in the output line.
- Define parameters as you would for a Totalize (TOT) statement (see TOT in this section).

- Place the variable defined for the subtotal field in the output area.
- To provide a space for an asterisk whenever nonnumeric values are encountered during subtotaling, a space is forced in the output line after any variable created by an arithmetic operator. To eliminate the space, use the * option.
- Even though you can predefine the size of a variable to load in a SUB statement, you can also initialize the variable to match the size of a corresponding report field. This is especially useful with a named field because the name does not directly specify the field size; it also allows the SUB statement to adjust to a change in the size of a field.

Example 1: Subtotaling Subgroups

Subtotal the subgroups in the Cust Code field and add the values in the Ord Qty field:

```
@sub,0,d,1 '' 'custcode','ordqty' □,s,+ v1h,v2i .
v1 v2
```

where:

`''`	Uses no options
`'custcode'` `s`	Subtotals on the subgroups in the Cust Code field
`'ordqty'` `+`	Adds the values in the Ord Qty field
`v1h`	Initializes v1 as a type H variable to the size of the Cust Code field, to capture the name of the subgroup
`v2i`	Initializes v2 as a type I variable to the size of the Ord Qty field, to capture the subtotals for each subgroup

Example 2: Counting Entries and Omitting Error Flag

Count entries in report 1C0:

```
@sub,0,c,1 e* 12-5,33-8 □,s,+ v1i,v2h,v3i6 .
v2 v1 v3
```

where:

e*	Uses the E option to count the entries and the * option to omit the error flag
12-5 **s**	Subtotals on the subgroups in the Sub Key field (column 12 for five characters)
33-8 **+**	Adds the values in the Retail $$$$ field (column 33 for eight characters)
□	Processes tab lines only
v1i	Captures the entry count for each subgroup in v1
v2h	Captures the character from the Sub Key field in v2
v3i6	Captures the subtotals for each subgroup in v3
v2 v1 v3	Places the data in the output area in this format

TOT (Totalize)

The Totalize (TOT) statement performs arithmetic and move operations on fields within a report or result (and creates a result if you do not specify variables to capture the resulting values).

Format

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

Following is a description of the fields and subfields:

Field	Description
c,d,r	Report to process.
lab	Label to go to if no data is found.
o	Options field (see Options).
cc	Column-character positions or names of the fields to process.
ltyp	Line type to process (if you specify the A option, you can leave this subfield blank, but enter the comma).
p	Parameters: + Add (or multiplicand for multiplying) - Subtract * Multiply (multiplier) / Divide = Move to this field A Average C Cumulate D Delete commas from results E Count entries when using the S option I Insert commas in the results M Move this field S Subtotal S1-S5 Hierarchical Subtotals
vrslts	Variables to capture the totals (if you use this field, the TOT function does not create a result).

Options

A Processes all line types.

C Displays a subtotal if the value in the subtotal key field changes case. For example, with the C option, if the value in the key field is Rate on the third line and RATe on the fourth, a subtotal is shown after the third line. Use this option with the S parameter only.

1100: On OS 1100 MAPPER Systems, this option applies only to full character set (FCS) reports.

E Counts entries (see also the R option). In the TOT run statement, the first variable contains the entry count.

H Cumulates horizontally.

I Ignores headings: processes the report even though asterisk lines or the heading divider line may not exist.

J(*x*) Justifies numeric result values in fields to *x*; *x* is one of the values listed below. Leading zeros appear to the left of a value, nonsignificant zeros to the right. Neither affect the value.

- C Eliminates leading and nonsignificant zeros. Inserts commas every three digits in the integer portion of the result, if the field is large enough. Right-justifies values.
- L Eliminates leading and nonsignificant zeros. Left-justifies values.
- R Eliminates leading and nonsignificant zeros. Right-justifies values.
- X Eliminates leading zeros, left-justifies values, and fills fields with nonsignificant zeros.
- Z Eliminates leading and nonsignificant zeros, right-justifies values, and fills leading spaces with zeros.

N Omits the grand total.

O Omits all data lines from the result, displaying only subtotal and grand total information.

R*n*	Rounds numbers to the nearest *n*. Use with the E option to round entry counts. The range for *n* is .0000000000000001 to 100000. For example if you type r.001, the values are rounded to the nearest thousandth. If you type r10000, the values are rounded to the nearest 10 thousand.
S	Places subtotals in vertical operation fields.
=*x*	Changes the line type indicator in column 1 to *x*, where *x* is an asterisk, period, or alphanumeric character.
*	Omits asterisk flags following results of vertical operations, except for grand totals, when some of the data was invalid (nonnumeric or blank).

Example 1: Adding a Field and Capturing Entry Count and Total

Add a field in report 1C0:

```
@tot,0,c,1 e 42-7 □,+ <count>i4,<total>i7 .
```

where:

`e`	Uses the E option to count entries
`42-7` `+`	Adds the Sales Commiss field (42 for seven characters)
`<count>i4`	Captures the entry count in <count>
`<total>i7`	Captures the total in <total>

Example 2: Multiplying Two Fields

Multiply the Spaces Req field by the Demo Quantity field, place the product in the Demo Results field, and display the grand total of the Demo Results field at the end of the result:

```
@tot,0,c,1 '' 'space','demoquantity','demores' \
□,+,*,=/□,,,+ .
```

where:

`''`	Uses no options
`'space'` `+`	Multiplies the Spaces Req field (column 50 for five characters)
`'demoquantity'` `*`	by the Demo Quantity field (column 56 for eight characters)
`'demores'` `=`	and places the product in the Demo Results field (column 65 for 15 characters)
`/□,,,+`	Vertically sums the Demo Results field and displays the grand total at the end of the result
`□`	Processes tab lines only

Using the ITOT Run to Create a TOT Statement

This section on using the ITOT run applies only to the OS 1100 MAPPER System.

You can use the Iterative Totalize (ITOT) run to create a TOT statement. For information on how to use the iterative runs, see the *Manual Functions Reference*.

UNX (UNIX Interface)

This section applies only to the U Series MAPPER System and UNIX MAPPER Systems.

The UNIX Interface (UNX) statement executes UNIX commands.

Format

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

Following is a description of the fields and subfields:

Field	Description
c,d	Drawer into which the result is to be returned (-f option).
logon	UNIX logon. Default = UNIX logon of the current run user.
pw	UNIX logon password. Default = UNIX logon password of the current run user.
lab	Label to go to if the UNIX shell cannot be started.
o	Options field. Enter options in lowercase characters. If you do not specify options, you must designate the options field with two apostrophes (' ').
cmd	UNIX command to be performed. Standard UNIX searching is done to find the command. If the path is defined in your shell variable PATH, you do not need to enter the full path name. Enclose commands containing embedded spaces in apostrophes ('). If you do not specify a command, designate its position with two apostrophes (' ').
args	Arguments passed to the UNIX command. Enter full path names only if the command requires them. When entering arguments, enclose the command and argument in apostrophes.

Options

-c Clears the screen after the statement finishes processing and returns to the MAPPER system.

-f Returns the output of the command as a result in drawer A. If you enter a file name immediately following the -f option, the function places a copy of the output from the UNIX command into that file. Do not use the -f option when executing an interactive UNIX process such as vi and ed. (Do not use this option with the -w option.)

-p Executes the user's .profile before executing the UNIX command.

-w Allows the run user to press **Resume** to resume the run once the command is executed and the output has been displayed. (Do not use this option with the -f option.)

Comments

- If the run user entered the MAPPER system through the UNIX shell, the user can execute any UNIX commands normally available through that logon. When the UNIX command completes, the MAPPER run continues.
- If the run user entered the MAPPER system directly from the UNIX logon prompt (with a logon of **mapper**), the run terminates with an error. (The mapper logon allows access only to MAPPER software.)

  To prevent the run from terminating, be sure the UNX statement contains a valid logon and password for all users of the run. You may want to prompt the user for a valid logon and password before executing the UNX statement.

  When the command completes, the MAPPER run continues.
- If you use UNIX statements in runs that are to be executed in the background from a remote site (RRN, NRN, NRM, BPRUN$ command with the STR statement), you must specify a valid UNIX logon and password for the remote site.

- You may want to use the Refresh Screen (PNT) statement following a UNIX command to reestablish screen attributes when the run user returns from the UNIX environment.
- UNIX commands, arguments, and file names are case sensitive. For example, the command /BIN/LS -L does not execute the /bin/ls -l command.

Examples

Access the UNIX system shell; the ***logon*** and ***pw*** fields are specified by <runlogon> and <runpw>:

```
@unx,,<runlogon>,<runpw> '' '' .
```

Note: *You must preserve the positions of options and commands.*

The user can return to the MAPPER system by pressing **Exit**.

List the contents of the /usr/work/newuser directory and place the output in a cabinet 0, drawer B result:

```
@unx,0,b -f '/bin/ls -l /usr/work/newuser' .
```

Execute the UNIX cat command on file prod_status, and put the output in file tmpfile and in the cabinet 0, drawer E result:

```
@unx,0,e -f/tmpfile '/bin/cat '\
'/usr/work/newuser/prod_status' .
```

USE (Use Variable Name)

The Use Variable Name (USE) statement assigns a variable name to a specific variable number. See Section 4 for a description of named variables.

Format

```
@USE name=v[,name=v,...,name=v] .
```

Following is a description of the subfields:

Field	Description
name	Name to associate with the variable.
v	Numbered variables to which to assign a name. If you want to initialize the variable, include the type and size (such as v2h12).

Comments

- Normally, when you assign a name to a variable, you can refer to it only by the name and not a variable number. However, if you assign the variable with the USE statement, you can refer to it by either its name or number.
- As a rule, you should not use named and numbered variables within the same run. There may be certain cases, however, in which it is necessary. For example, you may need to call an external subroutine containing numbered variables from a run that uses named variables.
- You can assign several names with a single USE statement and optionally initialize the variables at the same time. If you want to initialize the variable while naming it, include the variable type and size with the variable number.

Examples

Assign the name <station> to v190:

```
@use station=v190i5 .
```

Assign the name <designer> to v191:

```
@use designer=v191s77 .
```

WAT (Wait)

The Wait (WAT) statement temporarily suspends the execution of a run.

Format

```
@WAT[,M,lab] ms .
```

1100:

```
@WAT ms .
```

Following is a description of the field and subfields:

Field	Description
1100: The **M** option and the *lab* subfield are not available on OS 1100 MAPPER Systems.	
M	Option that suspends the run until an outstanding message is received or until the time specified by *ms* elapses.
lab	Label to go to if an outstanding message is received.
ms	Time in milliseconds to suspend the run. Specify an integer from 0 to 600,000. (600,000 milliseconds equal 10 minutes.) 1100: On OS 1100 MAPPER Systems, specify an integer from 0 to 60,000. (60,000 milliseconds equal 60 seconds.)

Comments

- The M option on the WAT statement allows the run to stall until an outstanding message is received or until the wait time is reached. If a message is received and a label is specified, the run continues at the label. If a message is not received within the wait time limit specified, the run continues at the next statement.

1100: The M option is not available on OS 1100 MAPPER Systems.

- Use WAT statements interspersed with extensive logic loops to reduce the impact on the MAPPER system.

- You can also use a WAT statement to hold an interim display (DSM, DSP, OUV, and OUT) on the screen long enough to read it.
- Do not place any run statements on the same line following the WAT statement. The logic scan of the line terminates after executing the WAT statement.

Examples

The run waits 100 milliseconds before resuming:

```
@wat 100 .
```

1100: The following example does not apply to OS 1100 MAPPER Systems.

This example sends a report to <user>, uses the WAT statement with the M option to wait for the user to receive the message, and displays status messages on the run user's screen:

```
@020: . Send Report
@     snu,0,b,2,<user>,<dept>,<site>,y,199 .
@     wat,m,099 500000 . **Wait for user to receive msg***
@     ouv,1,1 ers$ .
@     ouv,15,15 'Message not received' .
@     wat 4000 . ***Suspend OUV display****
@     rel .
@099: . Process Response
@      .
@      . (other processing based on response)
@     rel .
@199: . Error Routine
@     ouv,1,1 ers$ .
@     ouv,15,15 'Problem with user-id, dept, or site' .
@     wat 4000 . ***Suspend OUV display***
@     rel .
```

WRL (Write Line)

The Write Line (WRL) statement updates up to 23 lines of a report. The WRL statement updates the report, renames that report -0, and releases any previous -0 result.

Condition

Precede all WRL statements with an Update Lock (LOK) statement unless you are processing a result. Enter an Unlock (ULK) statement following the WRL statement to release update control.

Format

```
@WRL,c,d,r,l[,ntuid?,wpw,ccpy?] cc ltyp,vld .
```

Following is a description of the field and subfields:

Field	Description
c,d,r	Report to update.
l	Line number to update.
ntuid?	Do not update the time and user-id in the date line, Y or N. Default = N. If you specify Y and the update date on the report is not the current date, the system overrides your request and writes the update time, date, and user-id.
wpw	Write password to place on the report. Use this subfield to set a password on a report. Note that the fields after *wpw* are all mandatory; you cannot set a write password without updating the report.
1100: *ccpy?*	(This subfield applies only when using the Deferred Update [DFU] statement on the OS 1100 MAPPER System.) Write a complete copy of the report if this is the first WRL statement since executing the DFU statement, Y or N. Default = N. Writing a complete copy of the report may cost additional I/O overhead to update the report; however, it can considerably decrease the cost of other users attempting to access the same report while it is under a deferred update lock.

continued

continued

Field	Description
cc	Column-character positions or names of the fields in which to write data.
	1100: Note that starting with level 35R1, the WRL statement writes data on period lines in the columns you request in the *cc* subfield. In previous levels, the data would always be written starting in column 2.
ltyp	Line type designator to write. Updated lines will be changed to the line type you specify. (In other run statements, the *ltyp* subfield specifies the type of line to process; however, the *ltyp* subfield in the WRL statement specifies the line type to be written.)
vld	Content of variables, literals, reserved words, constants, or any combination of these, to write.

Examples

Use the LOK statement to lock report 3B0, for update, then use the WRL statement to write a tab line in line 6 of the report, placing the characters sh in column 2 for two characters. Use the ULK statement to release update control:

```
@lok,0,b,3 wrl,0,b,3,6 2-2 □,sh ulk .
```

For a multiline write, expand the parameters field, as in this example:

```
@lok,0,b,3 .
@wrl,0,b,3,6 5-6,77-3 □,900605,TUE/□,900613,WED .
@ulk .
```

This WRL statement writes two tab type lines (lines 6 and 7) in the report with 900605 (line 6) and 900613 (line 7) in column 5 for 6 characters, and TUE (line 6) and WED (line 7) in column 77 for 3 characters.

XQT (Execute)

The Execute (XQT) statement executes one or more run statements, including data for output areas. You can execute up to 132 characters of data.

1100: On OS 1100 MAPPER Systems, you can execute up to 640 characters of data.

Format

```
@XQT[,lab | vld] .
```

Following is a description of the field and subfield:

Field	Description
lab	Label or line number to execute.
vld	Variables, literals, constants, reserved words, or any combination of these, from which to formulate and execute a run statement.

Comments

- You can use the *lab* subfield to execute a line already containing one or more run statements, or you can formulate the statement to execute within your run, using variables, for example, containing data captured from an input screen that you want to include in a run statement.
- After interpreting the XQT statement, the system executes either the data at the specified label or line number or the data specified in the *vld* field.
- If the data you want to execute is a run statement, the first character of the data must be an at sign (@); otherwise it is treated as data for the output area and the run continues at the next line in the run control report. (The at sign [@] tells the MAPPER interpreter that the information that follows is a run statement.)
- You can include any characters in the data you want to execute except the reverse slant (\ ). You need not enclose commas (,) or slants (/) in apostrophes.

- If the statement being executed errs, the run goes to the label given on the executing statement. If the executing statement does not contain a label, the run terminates in error with the XQT statement, and a system message appears.
- You can nest XQT statements. That is, one XQT statement can execute another XQT statement, which, in turn, can execute another XQT statement, and so on.
- After executing the XQT statement, the run continues execution at the line following the XQT statement, unless the executed statement transfers control to a new location (for example, through a GTO, RUN, or XIT statement).
- An XQT statement terminates the scan of the current line in the run control report.

Example 1: Reading and Executing a Line from Another Report

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

```
@rdl,edrw$,2,<line>,099 1-80 <data>s80 .
@xqt <data> .
```

Example 2: Executing a Line Computed from a Screen Menu Selection

Determine the line to execute from an input screen:

```
1. @out,-1,2,10,1,1,,,,5 .
2. @chg v1i2 curv$ +3 if v1 ge 3 & le 5 gto lin+1 ; \
     gto 001 .
3. @xqt,lin v1 .
4. @dsp,-0 .
5. @rel .
6. @srh,-0,,,099 dh 'status date'  ,861225 .
7. @srh,-0,,,099 dh 'ship date'    ,870614 .
8. @sor,-0 '' 'product type'    ,1 .
```

1. Display the menu to the user.
2. Capture, in v1, the user's menu selection based on the vertical cursor position and adjust it to form the bias to the line that will be executed to handle this menu selection.
3. Execute the relative line pointed to by CURV$+3.
4. Display the result of the user's selection.
5. Release the display if the user resumes the run.
6. Execute this line if the user presses **Transmit** from line 1 of the screen.
7. Execute this line if the user presses **Transmit** from line 2 of the screen.
8. Execute this line if the user presses **Transmit** from line 3 of the screen.

XUN (Exit MAPPER System)

This section does not apply to the OS 1100 MAPPER System or the Personal Computer MAPPER System.

The Exit MAPPER System (XUN) statement exits the user to the previous environment.

Format

```
@XUN .
```

Comments

- The XUN statement terminates the active run, signs off and terminates MAPPER software for that station, and exits to the previous environment.
- **U Series and UNIX:** The user receives a UNIX logon prompt if MAPPER software was executed with the mapper logon. If MAPPER software was executed from a UNIX shell, a UNIX shell prompt is displayed.

Example

Terminate the run and MAPPER software at the station and exit to the previous environment:

```
@xun .
```

*cmd (Local Code)

This section does not apply to the OS 1100 MAPPER System or the Personal Computer System.

The Local Code (***cmd**) statement executes a site-defined run statement. See Appendix G for more information.

Format

```
@*cmd .
```

Following is a description of the call:

Field	Description
cmd	Run function call of a site-defined local run statement.

Your site is responsible for documenting the name and format of each site-defined run statement. See your coordinator or the person responsible for maintaining the MAPPER system software and databases for information on the site-defined local run statements available for your use.

Glossary

A

A Series data file
A file formatted in such a way that it can be read by an A Series editor, such as Command and Edit language (CANDE).

abort
To terminate execution of a MAPPER function or run.

abort routine
A subroutine to be executed if a user aborts the run during execution. It can be an internal or external subroutine. *See also* external subroutine, internal subroutine, subroutine.

accesscode
An identification code subordinate to a usercode. An accesscode is used to further establish a user's identity, control security, and restrict access to disk files. *See also* log on.

accounting log
A summary of data containing a record of every transaction compiled while a run executes. The Log for Analysis (LOG) run statement produces an entry in the accounting log for each function executed in the run. *Synonym for* log list.

active screen
The screen that shows you have signed on. Although the active screen looks like the sign-on screen, your user-id takes the place of the word Idle, and the bottom line indicates which cabinet you are accessing. *Contrast with* sign-on screen. *See also* caret, sign on.

alphanumeric variable
A variable that contains one or more alphabetic, numeric, and special characters, such as the tab character. *See also* variable, variable type.

American Standard Code for Information Interchange (ASCII)
A set of numeric codes that defines a character set. The ASCII character set used in MAPPER software is called full character set (FCS). *See also* character set, full character set.

application
See MAPPER application.

apostrophe
A special character (') used in run statements to delimit literal data that contains spaces, slants, or commas. *See also* run statement.

argument
An item of data passed to a computer program. *See also* parameter.

arithmetic expression
A single numeric value or a combination of two or more values and one or more arithmetic operators. *See also* arithmetic operator.

arithmetic operator
A special character (such as +, -, *, and =) that specifies a mathematical relationship between two values. *See also* arithmetic expression, expression.

ASCII
See American Standard Code for Information Interchange.

asterisk line
A line beginning with an asterisk (*) in column 1; it can be used as a comment line. It is not controlled by the tab positions and input edit codes of report 0. It has these characteristics: it can be shifted, it can extend to 256 characters, and it can be displayed and processed in different formats. *See also* line type.

auxiliary device
A peripheral device connected to a terminal, such as a printer. *See also* peripheral device.

B

B20 operating system (BTOS)
The operating system used when running INFOVIEW II applications on Unisys B20 series systems. *See also* INFOVIEW II.

background run
A MAPPER run that frees your terminal for other uses while it executes.

basic format
The unshifted columns of a report as defined in report 0 from left to right. On 80-character screens, the basic format contains the leftmost 80 characters of a report; on 132-character screens, the basic format includes the leftmost 132 characters of a report.

binary find process
A method of finding an item quickly in a sorted list. Instead of scanning data line by line, a binary find process samples the data at midpoint and continues dividing and sampling the data until it finds the target item.

binary synchronous communications (BSC)
A communications protocol for sending data to an IBM host computer.

Boolean logic
A system of logical comparisons named after mathematician George Boole. Boolean logic uses relational operators to evaluate expressions as true or false. *See also* conditional statement, IF/THEN/ELSE statement, relational expression, relational operator.

branch
To bypass the usual sequence of run statements and jump forward or backward from the current location in a run. *See also* conditional branching, label.

breakpoint
A place in a run, specified by a Run Debug (RDB) command, at which the run execution is interrupted for debugging purposes.

BSC
See binary synchronous communications.

BTOS
See B20 operating system.

BTOS II
The Unisys workstation operating system for the B2x and B3x series workstations.

C

cabinet
A group of eight drawers (B through I), referred to by number and usually used by a department. Each user signs on into a specific cabinet, assigned by the MAPPER system coordinator. Users in a particular department may have one or more cabinets to work in. *See also* department, drawer.

cabinet password
See password.

calculator
A feature of MAPPER software used by the Arithmetic (A) function that allows you to perform calculations. It can be used with a report containing predefined equations or independently, similar to a pocket calculator.

call
The abbreviation you use to specify a function, run, or run statement. For example, CAL is the call for the Calculate function.

CANDE
See Command and Edit language.

caret
A special character (^) that releases a displayed report or message, then displays the MAPPER active screen. *See also* active screen.

case sensitivity
Pertaining to the differentiation between uppercase and lowercase letters. When the option that determines case sensitivity is used, an uppercase letter is treated as a different character than its lowercase letter.

character hierarchy
The relationship of characters to one another. For example, in limited character set, numeric characters have higher values than alphabetic characters. In sort or search-in-range processes, alphabetic characters come before numeric characters.

character set
The characters allowed in the reports in a drawer. *See also* American Standard Code for Information Interchange, Fieldata, full character set, full character set upper, limited character set.

character string
A series or group of connected characters.

chart input report
A report that you fill in with data you want to chart. After you have finished filling in the chart input report, it is called a completed input report. *See also* completed input report.

CMS 1100
See Communications Management System.

COBOL
See Common Business Oriented Language.

column
A character position in a report; for example, the first character position on the left side of a report is column 1.

column-characters (*cc*) field
The part of a run statement that specifies which fields in a report are used by that run statement. For example, column-characters field 15-9 is the field that starts in column 15 and is nine characters long. Note that field names may be used in place of the column-characters field of a run statement. *See also* field name.

column-formatted report
A report having a layout identical to other reports in the same drawer. The fields are separated by tab characters. *Contrast with* freeform report.

command
An instruction entered into a report to carry out operations with a function, such as GOC commands processed by the Generate Organization Chart (GOC) function, or data control commands used with the Create File (FILE) function or Create File (FIL) run statement.

Command and Edit language (CANDE)
A message control system (MCS) that prepares and updates files in an interactive, terminal-oriented environment. *See also* message control system.

comment
The part of a run statement that follows the space-period-space sequence and summarizes what the run statement or subroutine is doing at that point in the run. It documents the run for later reference.

Common Business Oriented Language (COBOL)
A high-level computer language used mainly in the business environment.

Communications Management System (CMS 1100)
The communications software product that provides the structure for all communications and network capabilities for OS 1100. CMS 1100 software allows a site to establish remote connections from an OS 1100 system to other hosts, terminals, and networks.

Communications Management System (COMS)
A general message control system (MCS) that supports a network of users and provides them with a consistent online interface to the host system. *See also* message control system, online.

communications output printer (COP)
Any type of printing device connected to a terminal.

completed input report
A report that contains data to be charted by a specific color graphics run, such as PIEG. *See also* chart input report.

COMS
See Communications Management System.

concatenated variables
Variables that are joined together in a specific sequence. *See also* variable.

conditional branching
A method of controlling the sequence in which run statements are executed. At the branching point, a specific factor (such as the value of a variable or reserved word) determines which statement is executed next. *See also* branch.

conditional statement
(1) A statement that sets up a condition to determine the subsequent functions a run performs. (2) A statement of the Calculate (CAL) run statement that tests expressions and controls processing of individual equations, depending on whether a specified condition is true or false. The four conditional statements are IF:, THEN:, ELSE:, and FIRST:. *See also* Boolean logic, IF/THEN/ELSE statement.

console
A display terminal used to execute functions for an operating system.

constant label
A value label that has a predefined value, such as PI. It is used in mathematical run statements such as the Calculate (CAL) run statement. *See also* field label, value label.

control characters
The characters entered in a run control report to control the output area displayed by the run. They are used to generate four-to-one and five-to-one output with the Output (OUT) run statement. These characters control the kind of data that the user enters in fields of the screen presented by the run. *See also* five-to-one output, four-to-one output.

control line
The top line of the screen containing control positions. *See also* control position.

control line procedure
The method of executing a function in which you type the function format on the control line and transmit. *Contrast with* menu procedure. *See also* control line.

control position
The position on the top line of the screen following the SOE character (♦) after the Line and Roll fields. *See also* control line.

coordinator
See MAPPER system coordinator.

COP
See communications output printer.

counter
A tally number, generated by a run, that records the number of times an event occurs. It is commonly used to escape a loop after a certain value is reached.

cumulation
The process of adding a quantity to a total and saving the new total in another field, repeated a number of times. For example, the Totalize (TOT) run statement can be used to cumulate a field. *See also* subcumulation.

current position
The point on the screen from which your next command originates when using graphics primitive code and expanded syntax commands. *See also* expanded syntax commands, graphics primitive code.

current result
The most recent result, called -0 (minus zero), when using run statements. *See also* renamed result, result, temporary result.

cursor
The character (▮) on the screen that can be moved anywhere to show where to enter data. The cursor shows your current location on the screen.

D

data line
Any report line below the heading divider line. *See also* line type.

data name
The name of a cabinet, drawer, or report that has been defined by using the NAME run. This name is stored in the system directory. *See also* system directory.

Data Transfer Module (DTM)
A software package that provides a fast, flexible method of transferring data between applications on OS 1100 computers.

data unit
A unit of related information consisting of a tab line and the following lines up to the next tab line. A data unit can start with other line types when using the Search function or run statement with the U option. *See also* paragraph.

database
The cabinets, drawers, and reports maintained in files by the MAPPER system.

date format
A format that defines how a date is to be displayed. For example, DD MMM YY displays a date as 06 JUN 90.

date input specification
The code to specify the format of a date to process.

date line
The first line (line 1) of each report, showing the date and time of the last update, report number, report creation date, and user-id of the last person to update the report. The date line is a period line and is counted as one of the heading lines. *Synonym for* line 1.

date output specification
The code used to represent the format of a result date.

DDP
See distributed data processing.

debug
To test a run and solve errors in the run statements or in the logic of the run.

decode
To translate an encoded report into a readable report, using the Decode Report (DCR) run statement. *See also* encode.

decrement
To decrease the value of a variable by an integer amount.

default
A preset value or condition that the system uses whenever you do not choose a specific selection.

default MAPPER site
The MAPPER site installed first on a host computer. *See also* site.

delimiter
A character placed on each side of a character string to indicate the beginning and ending of the character string. The delimiter can be any character other than those in the character string.

demand program
A program executed in demand mode, which processes data as quickly as it becomes available or ready.

demonstration database
The reports, usually in cabinet 0, in which you can practice manual functions and run statements.

department
A group of MAPPER system users specified by a number. You specify your department number when you sign on to the MAPPER system. *See also* user registration.

device
See peripheral device.

disk pack
A disk that consists of multiple platters stacked vertically on a central spindle. Data on a disk pack are accessed by movable read/write heads. Some disk packs are removable. *Synonym for* pack. *See also* pack family.

display
(1) The terminal screen you look at while using MAPPER software. (2) To present data on the screen.

distributed data processing (DDP)
A network consisting of two or more computers at different locations. These computers are connected by data communications and are capable of interfacing with each other to perform a job, task, or application.

distributed MAPPER application
An application that uses runs or data residing on more than one MAPPER system. *See also* MAPPER application.

downline load
To copy a file or other information from a computer system to your own terminal.

drawer
A group of reports in a cabinet. All reports within a drawer have the same headings and line length. Each drawer of a cabinet is identified by a letter from B to I. Drawer A is accessible to all cabinets in a MAPPER system. *See also* cabinet, freeform drawer, report 0.

drawer letter
A letter (A through I) used to identify a drawer in a cabinet. *See also* cabinet, drawer.

drawer number
The octal number that identifies the drawer and cabinet within the system. Each drawer has a unique number, such as 0010 for drawer E in cabinet 0. *See also* cabinet, drawer.

drawer password
See password.

DTM
See Data Transfer Module.

E

ECL
See Executive Control Language.

edit function
A function, such as SOE Update and Add Line, used to alter report lines.

empty report
A report that contains headings but no data.

encode
To transform a report into code using the Encode Report (ECR) run statement, making it unreadable unless the correct key is specified. *See also* decode.

end report line
The last line of a displayed report or result.

equation
Two arithmetic expressions separated by an equal sign (=).

equation operator
See operator.

equation set
The function mask and equations used with the Iterative Calculate (ICAL) run. *See also* function mask.

error message
See system message.

error routine
A subroutine to be executed if a run encounters an error. It can be an internal or external subroutine. *See also* external subroutine, internal subroutine, subroutine.

Exec
See Executive system.

execute
To process a particular instruction or task. Manual functions, runs, and run statements can be executed.

Executive Control Language (ECL)
The computer language used to communicate with the Executive system (Exec). *See also* Executive system.

Executive system (Exec)
A program that controls the execution of other routines. As the operating system, the Exec is the principal interface between the user and the system as a whole. *See also* Executive Control Language.

expanded syntax commands
A set of graphics commands translated by the Graphics Scaler (GS) function or run statement into graphics primitive code to create charts or modify existing charts. *See also* graphics primitive code.

exponential notation
The use of symbols to indicate the number of places to move the decimal. For example, 12000 in exponential notation is 12E3.

expression
A series of one or more field labels, values, or operators that produces a single arithmetic or logical value. For example, in the Calculate (CAL) function, an expression is the sequence of field labels and arithmetic operators (+,-,*, or /) that completes an equation. *See also* arithmetic expression, relational expression, relational operator.

external subroutine
A subroutine in another run control report to which your run temporarily transfers control. *Contrast with* internal subroutine. *See also* abort routine, error routine, subroutine.

F

fast access
A form of function or run requests that allows you to bypass the function forms or menus. *See also* function form, menu.

FCS
See full character set.

FCSU
See full character set upper.

field

(1) A series of one or more columns of a report that are defined as an entity, such as a status code or a shipping date. (2) A selection from a menu or a position in a system message where data is entered (for example, the **Report** field in a function form). (3) A defined part of a run statement format. Fields are separated from one another by a space. *See also* subfield.

field headings

The column headings for report fields. These titles appear above each field at the beginning of a report, immediately preceding the heading divider line. *Synonym for* headings. *See also* heading divider line.

field label

A single alphabetic character that identifies a single field or several fields for equations in mathematical functions or run statements. *See also* constant label, value label.

field name

A name that identifies a field of a report. The name is derived either from the two heading lines preceding the heading divider line of the report or from report 0 if the entire drawer is being processed. *See also* heading divider line, report 0.

Fieldata

A set of codes that define a character set. The Fieldata character set used in MAPPER software is called limited character set (LCS). *Contrast with* American Standard Code for Information Interchange, full character set. *See also* character set, limited character set.

five-to-one output

Screen output produced by lines in a run that use control characters and emphasis characters to control special editing and presentation effects. Five lines in the run are used to define one line of output. Five-to-one output is used with color displays; four-to-one output is used with monochrome displays. *See also* control characters, four-to-one output.

flowchart
A diagram of procedures, subroutines, and branches used to plan a run.

form
A report containing screen control commands. *See also* screen control commands.

format
(1) One of many variations of a report within a drawer (for example, basic format, format 1, format 2, and so on), each of which displays a different selection of columns of data in the report. The formats are defined in the report 0 of that drawer. (2) The specific fields and subfields used in run statements, which vary from one statement to another. *See also* report 0.

format lines
The lines in report 0 of a drawer that specify the report columns displayed in many different report formats. *See also* drawer, format, report 0.

format sensitive
Pertaining to a run that processes a displayed report in a specific format. The run must be registered as format sensitive. *See also* format.

four-to-one output
Screen output produced by lines in a run that use control characters and emphasis characters for special editing and presentation effects. Four lines in the run are used to define one line of output. Four-to-one output is used with monochrome displays; five-to-one output is used with color displays. *See also* control characters, five-to-one output.

freeform drawer
A drawer used for freeform reports. Drawer A is a freeform drawer accessible from all cabinets in the system. *See also* drawer, freeform report.

freeform report
A report without a columnar structure determined by the report 0 of its drawer. You can use freeform reports for memos, bulletins, run control reports, or informal columnar reports. *Contrast with* column-formatted report. *See also* report 0.

full character set (FCS)
The character set that allows uppercase and lowercase letters, stored internally as ASCII characters. *Contrast with* Fieldata, limited character set. *See also* American Standard Code for Information Interchange, character set, full character set upper.

full character set upper (FCSU)
The set of ASCII codes in uppercase only. *See also* American Standard Code for Information Interchange, character set, full character set.

function
An operation you perform on one or more reports or on a result. Examples of functions are Match (MA), Search (S), or Sort.

function bar
An information bar at the bottom of the MAPPER active screen. The ten key names correspond to the **F1** to **F10** function keys. *See also* function keys.

function call
See manual function call, run function call.

function form
A screen, resembling a menu, that contains fields in which you supply information needed to perform a manual function. *See also* fast access, menu.

function keys
The set of keys on your terminal keyboard (for example, **F1** or **F2**) programmed to perform operations when you press them. *See also* function bar.

function mask
A screen of field headings for a report; you enter options above the field headings and parameters below the headings as instructions for a manual function. *See also* parameter.

G

granularity
A measurement used when allocating storage space to a mass storage file. For example, the Element (ELT) statement assigns a file with a maximum granularity of 262,143 tracks. *See also* mass storage file.

graphics primitive code
A set of commands that allows the MAPPER system to display charts on different types of output devices. You can design your own graphics by entering graphics primitive code in a freeform report. Graphics primitive code can be stored in MAPPER reports. *See also* expanded syntax commands, packed graphics primitive code, unpacked graphics primitive code.

H

heading divider line
The line beginning with an asterisk and made up of equal signs and periods that separates the field headings from the data; counted as one of the heading lines. Many functions rely on the heading divider line to determine where field headings end and data begins. *See also* field headings, field name, heading lines.

heading lines
The lines including and following the date line that show the drawer letter and report number, the names of fields, and the heading divider line. *Synonym for* report headings.

headings
Synonymous with field headings.

HELP run
A MAPPER run that supplies online information about functions, run statements, system messages, and other MAPPER operations. *See also* online, run target.

Hollerith variable
A variable that can hold alphabetic, numeric, or special characters. *See also* variable, variable type.

home position
The upper left corner of the screen.

horizontal operation
A computation performed in one or more fields across each data line in a report. *See also* vertical operation.

HOST1100 run
A run residing on the host OS 1100 MAPPER System that controls data transfer between the OS 1100 MAPPER System and the U Series MAPPER System. *See also* INSTALL1100 run.

host computer
The computer that holds the database and software of a particular MAPPER site. *See also* site configuration.

I

I/O
See input/output.

I/O request
A request for input from or output to system storage. *Synonym for* I/O.

IF/THEN/ELSE statement
A conditional statement that controls further processing based on the value of the expression in the IF: statement. If that expression is true, the expression in the THEN: statement is processed. If the expression is false, then the ELSE: expression is processed. *See also* Boolean logic, conditional statement.

increment
To increase the numeric value of a variable by an integer amount.

INFOVIEW II
A sophisticated and flexible software tool that allows a workstation to interface to a host computer. Your PC or B20 workstation uses INFOVIEW II to access the A Series MAPPER System. *See also* B20 operating system.

initialize
To assign a type, size, and initial value to a variable. *See also* variable.

input edit code
A numeric code (0 through 9) in report 0 of each drawer specifying what kind of data is allowed in each character position of the report. The person designing the drawer enters the input edit codes on the input edit line of the experimental report. *See also* report 0.

input/output (I/O)
(1) Information coming from or going to system storage. (2) An operation in which the system reads data from or writes data to a peripheral device such as a disk drive.

input parameters
The variables, literal data, reserved words, or any combination of these, to be processed in a MAPPER run. *See also* literal representation, reserved word, variable.

INSTALL1100 run
A run provided with the U Series MAPPER System that sends the complete HOST1100 run to the OS 1100 MAPPER System and replaces the temporary HOST1100 reception run. *See also* HOST1100 run.

integer
A whole number with no fraction or decimal part. For example, the numbers 1 and 2.

integer variable
A variable that contains integer numbers. *See also* integer, variable.

interactive
Pertaining to a process that can take place during another process.

interim display
Information displayed on the screen for a given length of time; the run then continues automatically.

internal subroutine
A subroutine within your run control report to which your run temporarily transfers control. *Contrast with* external subroutine. *See also* abort routine, error routine, subroutine.

issuing report
The report from which data is taken when using a run statement that processes two reports, such as the Append Report (ADD) or Match (MCH) run statements. *Contrast with* receiving report.

J

JDOE database
See demonstration database.

job
A group of one or more tasks, usually processed from and under the control of a single program. A job is assigned a number by the system and treated as a discrete unit of work by the computer.

justify
To position data within a field, variable, or report. For example, if data in a field is left-justified, it begins in the leftmost column of that field.

K

Kanji characters
A character set used by several Asian languages, including Japanese and Chinese.

known trailing substring
A substring in which you specify the starting character position and use the remaining characters in the field. For example, V1(2-0) specifies the substring beginning with the second character through the end of V1. *Contrast with* unknown trailing substring. *See also* substring.

L

label
(1) A number, preceded by an at sign (@) and followed by a colon (:), used to identify a run statement line. Labels organize sections within a run, allowing branching. (2) A name for a data value in an arithmetic expression. *See also* branch, label table definition lines.

label table definition lines
The lines at the beginning of a run that indicate the location of each label in the run. *See also* label.

LCS
See limited character set.

limited character set (LCS)
The character set that allows uppercase letters only, stored internally as Fieldata characters. *Contrast with* American Standard Code for Information Interchange, full character set. *See also* character set, Fieldata.

line 0
The first line of a report, never visible on the screen, containing system information about the report: the drawer letter, report number, write password, read password, number of heading lines, language of the report, and number of lines of the report. This information is stored with the report and can be displayed with the Line Zero (LZ) function or LZR run statement.

line 1
Synonymous with date line.

line type
A type of data line in the MAPPER database, specified by a line type designator in column 1. *See also* data line, line type designator.

line type designator
A character in column 1 of a report line that specifies the line type. There are four designators, indicating four line types: tab (□) — column-formatted, edited line (□ represents a tab character); asterisk (*) — column-formatted, nonedited line; period (.) — comment, nonedited line; and special (any valid character) — column-formatted edited line (may not start with a tab character, asterisk, or period). *See also* line type.

literal representation
The explicit and actual value of an item; information is interpreted exactly as it appears. For example, the literal representation of variable V11 is the characters V11, not the value that the variable holds.

LLP
See logic lines processed.

load
To initialize a variable with a value.

log list
Synonymous with accounting log.

log on
To access the A Series system by entering information that identifies you to the system. This data includes a valid usercode, password, and accesscode. Note that this is not the same as sign on, which is a procedure used to access the MAPPER system. *See also* accesscode, usercode.

logic lines processed (LLP)
The lines in a run control report processed during run execution. Each scan of a line counts as one LLP. *See also* run control report.

logic scan
The processing of each run statement line by the MAPPER system during execution of a run. *See also* run statement line.

logo
See active screen, sign-on screen.

loop
A sequence of run statements that execute repeatedly until a specified condition is met.

M

manual function
A MAPPER software command, such as Search (S) or Totalize (TOT).

manual function call
The abbreviation used to request a function (for example, LOC for the Locate function).

MAPER files
OS 1100 mass storage files that make up the system database.

MAPPER application
A set of related tasks accomplished by one or more runs and the database processed, for example, an inventory system. *See also* distributed MAPPER application.

MAPPER calculator
See calculator.

MAPPER format
The arrangement of data in a native system file that has a fixed line length and contains a transparent line 0, line numbers, and end-of-line control characters. The last line is the visible end report line.

MAPPER site
See site.

MAPPER software
The multiactivity real-time program that creates an end-user environment for file management and report generation. *See also* MAPPER system.

MAPPER system
A file management system that allows the user to maintain and manipulate a large amount of data in a report-structured database.

MAPPER system coordinator
The person who manages the database of the MAPPER system. This person configures the system, registers new users and new or updated runs, and coordinates the system user group.

MAPPER system logo
See active screen, sign-on screen.

MAPPER system operator
The person responsible for daily operations including mounting recovery tapes, performing the purge and cycle/merge processes, starting the MAPPER system, and executing the PREMAP or PRESTR runstreams. The system operator's responsibilities are specifically defined at each site.

MARC
See Menu-Assisted Resource Control.

mass storage
Peripheral devices that can store large amounts of data. *See also* peripheral device.

mass storage file
A file used as a secondary storage device that provides current report information on a specific program segment.

MCS
See message control system.

menu
A list of items on a screen from which one item can be selected, either by tabbing to the selection and transmitting, or by typing in a letter, number, or character string, and then transmitting. *See also* fast access.

Menu-Assisted Resource Control (MARC)
A menu-driven interface and transaction processor for users and operators of Unisys A Series systems.

menu procedure
The method of executing a function in which you select the function from a menu. *Contrast with* control line procedure. *See also* menu.

message control system (MCS)
A program that controls the flow of messages between workstations, application programs, and the operating system. *See also* Command and Edit language, Communications Management System.

message wait signal
A signal indicating that your workstation has a incoming message. When a message wait signal occurs, **MSG** is displayed at the upper right corner of the screen and the terminal beeper sounds. You press the **Message** key to display the incoming message.

multiple field label
A label that identifies more than one report field when using mathematical functions or run statements. *Contrast with* single field label.

N

named report
A report that has been named using the NAME run. For example, report 6B0 could have a name of Inventory.

named variable
A variable that is designated by a name rather than by a number. Greater than and less than characters are placed before and after the name, for example, <sum>. *Contrast with* numbered variable. *See also* variable.

nested statement
A run statement contained within another run statement. *See also* run statement.

nested subroutine
An internal subroutine referenced by another subroutine. Note that external subroutines cannot be nested. *See also* external subroutine, internal subroutine, subroutine.

NEWUSER database
See demonstration database.

nontab line
Synonomous with special line.

null
A blank character (not 0 and not a space) used to fill data lines.

numbered variable
A variable that is designated by the letter V followed by a number; for example, v1. *Contrast with* named variable. *See also* variable.

O

object code file
The executable code produced by a compiler.

online
Pertaining to the availability of a process or operation while another is in progress. For example, while using the Sort function, you may access online HELP.

operating system (OS)
A collection of computer programs that control the operation of a computer and peripherals.

operator
(1) A MAPPER system operator. (2) A symbol that specifies a calculation or comparison to be performed in a function such as Calculate (CAL). *See also* arithmetic operator, relational operator.

operator display terminal (ODT)
The system console device that allows the operator to enter commands directly to the operating system to perform various functions.

option
A selection that you can make for special operation of a manual function or run statement. If no option is used, the default operation occurs.

organization chart
A chart created with the Generate Organization Chart (GOC) function or run statement that shows how a company or other organization is structured.

OS 1100
See Executive system, operating system.

output area
A scratchpad or storage area used in a run that temporarily stores data collected during run execution. *See also* output lines.

output lines
The lines in a run control report that do not start with at signs (@) or colons (:). Output lines are placed in the output area and can be displayed when the run terminates. They can also be placed in a result by using the Break (BRK) or Break Graphics (BRG) statement and can be displayed or processed at any time during the run. *See also* output area, result.

P

pack
To eliminate leading and trailing spaces from a value held by a variable. *See also* disk pack, pack family.

pack family
A disk or a collection of disk packs on which physical files are stored. It is given a name of up to 17 alphanumeric characters assigned during installation and is generally designated for specific applications and users. *See also* disk pack.

packed graphics primitive code
One format for graphics primitive code in a result or report. In this format, the commands are strung together, not separated by spaces or other characters. *Contrast with* unpacked graphics primitive code. *See also* graphics primitive code.

packed variable
A variable in which the leading and trailing spaces have been deleted. *See also* variable.

paragraph
A group of data lines starting at either the type of line you are processing or the type of line specified with the Search U option. This includes subsequent lines up to, but not including, the next occurrence of the type of line processed or specified. *See also* data unit, line type.

parameter
An item of information supplied to a run statement to indicate specific values or fields to process. *See also* argument.

password
A character string used as a security feature to prevent unauthorized access to information on the MAPPER system. There are five kinds of passwords: the sign-on password restricts access to the MAPPER system; the write password restricts updates to reports; the read password restricts users from reading a report; the drawer password restricts access to specific drawers within a cabinet; and the cabinet password restricts access to a specific cabinet. *See also* user registration.

path
The logical course or line of direction taken by the system in the execution of a run or part of a run.

path name
The complete UNIX file identifier, including all directories, subdirectories, and the file name.

period line
A line beginning with a period in column 1; it can be used as a comment line. It is not controlled by the tab positions and input edit codes of report 0. It cannot be shifted, it is limited to the screen display size, and it cannot be displayed or processed in different formats. *See also* line type, save flag.

peripheral device
Any data communications or input device attached to a terminal or host computer. For example, an AUX printer is a peripheral device attached to a terminal; a modem is a peripheral device attached to the host computer. *See also* auxiliary device.

predefined lines
The lines in report 0 that contain tab positions, preset data, or reserved words. Each report 0 can have multiple predefined lines for which you can specify data or values that automatically appear when a new line is added to a report. These are data lines that have certain fields already filled in for permanent unchanged data and other fields left blank for entering changeable data. *See also* data line, report 0.

primitive graphics code
See graphics primitive code.

protected field
An area of a display terminal screen from which no input is accepted; the user cannot place the cursor within the area. Protected fields are defined by control characters using the Output (OUT) run statement. *See also* control characters.

Q

queue
To send output to an auxiliary device. For example, a report can be queued to a printer. *See also* requeue.

R

read password
See password.

receiving label
A name or alphabetic field label into which the result of a MAPPER operation is stored. For example, a generated count of records in a report could be assigned to the COUNT receiving label.

receiving report
The report to which data is sent when using a manual function or run statement that processes two reports, such as the Append Report (ADD) or Match (MCH) run statements. *Contrast with* issuing report.

receiving variable
The first variable in an equation and the variable that contains the result of a calculation.

registration
A procedure done by the MAPPER system coordinator to add new users or runs to the system.

relational expression
A sequence of operands and relational operators used to compare values or character strings. When evaluated, it produces a value of 1 if the comparison is true or 0 if the comparison is false. *See also* Boolean logic, expression, relational operator.

relational operator
A special character used to compare values, such as =, >, , <=, >=, or <> (equal to, greater than, less than, less than or equal to, greater than or equal to, and not equal to). *See also* Boolean logic, expression, relational expression.

relative line number
A line number specified by indicating the number of lines following or preceding the current line in a run control report. *See also* run control report.

remote file
A data communications file associated with any remote device attached to the system by way of a network communications processor. An object program that receives data from one or more remote devices and sends data to those devices regards the devices as a remote file.

remote run
A run that starts a run on another MAPPER system.

renamed result
A temporary result made by using the Rename (RNM) run statement to rename the current result (-0); for example, result -0 can be renamed to -1. These results are lost when the run ends or the terminal is released. *See also* current result, result, temporary result.

replacement string
A string of characters with which to replace the target string. *See also* target string.

report
The set of data that you work with in the MAPPER system. Reports are identified either by a unique report number or by a meaningful name you have given the report using the NAME run. *See also* report name.

report headings
Synonymous with heading lines.

report identifier (RID)
A specific report identified by a unique report number and drawer letter. For example, RID 2B refers to report 2 in drawer B.

report name
The characters used to refer to a report: either the report number and drawer letter or a meaningful name you have given the report using the NAME run. *See also* report.

report 0
A report that resides in each drawer in the MAPPER system. It serves as a template for the reports in its drawer when the Add Report function or run statement is used. *See also* drawer, format, function mask.

requeue
To send output already queued to an auxiliary device that failed to print the first time. *See also* queue.

reserved word
A character string reserved for specific use. The system supplies the information stored in the reserved word. For example, DATE1$ is a reserved word that supplies the current date in the format YYMMDD.

restricted access function
A function requiring special permission from the MAPPER system coordinator before it can be used. Applies to functions that require advanced knowledge of MAPPER software or the operating system.

result
A temporary copy of data obtained by executing a manual function or run. It is held in scratch storage until released, duplicated, or replaced into a permanent report. *See also* current result, renamed result, temporary result.

resume
To continue a function or run that was halted by a display. Press **Resume** to resume an operation.

retrieve
To bring a MAPPER report that has been filed into a native file back into the MAPPER database.

reverse slant
A special character (\ ) used to indicate that a run statement continues on the next line.

reverse video
Pertaining to highlighting on a screen where the background and character colors are reversed.

RID
See report identifier.

roll
To move vertically through the data in a report; to move a report forward or backward on the screen.

run
A series of instructions (run statements) stored in a run control report that the MAPPER system interprets to produce a report or perform other tasks, such as updating reports. *See also* run control report, run statement.

run call
Synonymous with run name.

run control report
A MAPPER report containing sequential run statements of step-by-step instructions for processing reports, results, or other data. *See also* run, run statement.

run function call
An abbreviation used to request a function in a run statement (for example, SRH for Search). *See also* run statement.

run name
A name assigned to a run; used to access the run and execute it (for example, HELP). *Synonym for* run call.

run registration
The process used by the MAPPER system coordinator to authorize and enable a run. The coordinator specifies in the run registration report the run name, the location of the run control report, and the person responsible for the run. *See also* run control report, run name.

run statement
A string composed of a run function call, options, fields, and subfields used to format an instruction that can be processed in a run. When the run statement is entered into a run control report, it can be processed to direct execution of MAPPER functions and other operations. *See also* run, run control report, run function call.

run statement format
See format.

run statement line
A line that contains one or more run statements, beginning with an at sign (@) and ending with a period. *See also* run statement.

run target
An item in a list of topics in the HELP run menu for run design. *See also* HELP run.

S

save flag
A date you place on line 2 starting at column 2 of a MAPPER report to prevent the report from being deleted before or on the date specified. Line 2 must be a period line in order to contain a save flag. Use the format @YYMMDD. *See also* period line.

screen control commands
A set of commands, used with the Screen Control (SC) statement, that perform basic screen operations, which include positioning the cursor, clearing the screen, defining fields, and defining screen attributes. These screen commands can also be used to design reports that define fields, box data in a specified area, and map function keys. *See also* form.

search information lines
The lines displayed at the top of a result, showing how many lines were found by the search, the total number of lines searched, and the search parameters.

semicolon
(1) A special character (;) used in run statements as field delimiters. (2) A special character used in Arithmetic (ART) and Calculate (CAL) statements to separate expressions. (3) A special character used in IF: statements to control more than one decision on the same line.

sign off
To terminate MAPPER software at your terminal. The sign-on screen appears on the screen. *Contrast with* sign on. *See also* sign-on screen.

sign on
To initiate MAPPER software at your terminal by entering data that identifies you to the system. This data includes your user-id, department number, and password (if applicable). The active screen appears on your screen. *Contrast with* sign off. *See also* active screen.

sign-on cabinet
The cabinet a user enters automatically when signing on to the MAPPER system and in which the user primarily works. It is set by the MAPPER system coordinator as part of user registration. *See also* user registration.

sign-on password
See password.

sign-on screen
The screen you see before signing on to the MAPPER system. It shows your system name, your station number, and the level of MAPPER software. The word Idle in your sign-on screen shows your station is inactive. *Contrast with* active screen. *See also* sign off.

single field label
A label that identifies one report field when using equations in mathematical functions and run statements. *Contrast with* multiple field label.

site
MAPPER software can support multiple databases referred to as sites. *See also* default MAPPER site.

site configuration
The devices (terminals, printers, or other MAPPER systems) connected to the host computer. *See also* host computer.

site letter
An alphabetic designator assigned to each MAPPER system.

slant
A special character (/) used to separate multiple parameters in a run statement.

SOE
See start-of-entry character.

special characters
The set of characters (such as] and *) on your keyboard that are not alphabetic or numeric characters.

special line
A line starting with any character except tab, asterisk, or period. It acts as a tab line although an alphabetic or numeric character appears in column 1. *Synonym for* nontab line. *See also* line type, line type designator, tab line.

stack
A data structure that stores variables on a last-in/first-out basis. As data is added, the stack moves down, with the last item added taking the top position. You can save (PSH), restore (PEK), restore and remove (POP), replace (POK), remove (RMV), and exchange (XCH) items from the stack.

start-of-entry (SOE) character
The character represented on the screen by this symbol: ♦. It is used with the SOE Update function and other edit functions. *See also* edit function.

statement
See run statement.

station
A terminal, workstation, or PC on the MAPPER system.

station number
The unique identifier for your terminal on the MAPPER system.

station sign-on screen
See sign-on screen.

string
See character string.

string variable
A variable that can contain up to 256 alphabetic, numeric, and special characters. *See also* variable, variable type.

subcumulation
To cumulate using the Totalize (TOT) run statement until a key field value changes so that the result yields two result fields. *See also* cumulation.

subfield
A part of a run statement field. For example, *ityp* is a subfield of the *ityp,p* field. Subfields are separated by a comma. *See also* field.

subroutine
A subset of a run that contains run statements performing a specific task within the run. *See also* abort routine, error routine, external subroutine, internal subroutine.

substring
A subset of characters within a character string. *See also* character string.

subtotal
A sum of data for groups of related lines.

system coordinator
See MAPPER system coordinator.

system directory
Reports that contain data names for cabinets, drawers, and reports. These are updated using the NAME run when adding, changing, or deleting a data name. *See also* data name.

system message
A message the MAPPER system displays on the top line of the screen to alert you to a possible or actual problem.

system logo
See active screen, sign-on screen.

T

tab

To move the cursor from one tab character position to another.

tab character

(1) A special character in MAPPER reports indicating a tab position on the screen. In this manual, a tab character on the screen is usually represented by a vertical bar (|) or center dot (·) but may be configured for your terminal as another character. (2) A special character used to specify tab lines. In this manual, a tab character is represented by a quadrate (□).

tab line

A line beginning with a tab character, usually a data line. It is controlled by the tab positions and input edit codes of report 0 and can extend up to 256 characters. *See also* input edit code, line type, line type designator, report 0.

target list

(1) A list of function calls in the HELP run. (2) A list of target words in an issuing report or list of words specified when you execute a function. The Word Locate (WL) and Word Change (WC) functions and run statements use target lists. *See also* HELP run, issuing report.

target string

The character string to be located or changed by a MAPPER function or run statement. *See also* character string, replacement string.

temporary result

A result created by a MAPPER function and given a new name using the Rename (RNM) run statement. The current result (-0) is renamed to -1, -2, and so on. A temporary result is released when the run ends. *See also* current result, renamed result, result.

time format

A format that defines how a time is to be displayed; for example, HHMMSS displays a time as 121500.

time input specifications
The codes used to define the format of a time to process.

time output specifications
The codes used to define the output format of a time.

TIP
See Transaction Processing.

toggle
To switch between two modes of operation using the same key. For example, pressing the key activates one mode; pressing the key again returns to the original mode.

trailer line
A line automatically attached to a data line when certain functions or run statements, such as the Totalize (TOT) function, are used. Asterisk and period type lines are trailer lines to tab lines. *See also* asterisk line, line type, period line, tab line.

Transaction Processing (TIP)
A part of the Executive system (Exec) used for real-time transaction processing.

transmit
To type information and press the **Transmit** key, sending the information to the host computer.

transparent character
A character that occupies a position in a string. When the string is compared to another character string, the transparent character allows all characters in that position to be accepted. For example, the $ in the character string A$C allows ABC and A1C to be accepted. *See also* character string.

truncate
To cut off or shorten; to exclude. For example, when using the Add On (ADON) function to add a report that has lines longer than those of the receiving report, the new lines are truncated.

U

Universal Terminal System (UTS)
A family of terminals and related hardware, for example, UTS 400.

UNIX
The operating system for the host computer that supports the U Series MAPPER System.

unknown trailing substring
A substring in which you specify the number of ending characters to use, regardless of the specific starting character position. For example, V1(0-2) specifies the last two characters of V1. *Contrast with* known trailing substring. *See also* substring.

unpacked graphics primitive code
One format for graphics primitive code in a result or report. In this format, each command is on a separate line. *Contrast with* packed graphics primitive code. *See also* graphics primitive code.

update control
The ability of MAPPER software to guarantee that only one user at a time may be updating a given report.

update password
See password.

update result
A result that can be used to replace or delete lines in the original report from which it was produced. *See also* result.

user input
Any information you are instructed to type. In this manual it is shown in color and in lowercase letters.

user registration
The process by which the MAPPER system coordinator enters into the user registration report a user-id, sign-on password, sign-on department number, sign-on language, and sign-on cabinet for each user. *See also* department, password, sign-on cabinet, user-id.

user-id
A unique code assigned to each MAPPER software user for security reasons. User-ids are set up by the MAPPER system coordinator to allow access to certain functions and runs. *See also* user registration.

usercode
An identification code used to establish user identity on the A Series system, control security, and provide for segregation of files. *See also* log on.

UTS
See Universal Terminal System.

V

value label
A name of up to six characters that identifies a single numeric or text value in mathematical functions or run statements. *See also* constant label, field label.

variable
A labeled entity (for example, V11 or <finds>) that can assume different values. Values can be assigned by the user or by MAPPER software. *See also* variable type.

variable table
A table created at the end of a run control report displaying the location of all variables in your run control report. You create a variable table with the Build Variable Table (BVT) run. *See also* run control report.

variable type
One of five types of variables. Each variable type specifies the kind of data the variable may hold. *See also* alphanumeric variable, Hollerith variable, integer variable, string variable, variable.

vertical bar
A special character (|) that represents a tab character in MAPPER reports. On some terminals, the tab character appears as a space. *See also* tab character.

vertical operation
A summary calculation (for example, sum or average) performed on a single field across all data lines. *See also* horizontal operation.

vertical operator
A special character (such as + or /) that identifies the type of arithmetic operation to be performed on all values in a specified field.

vertical summation
The process of adding fields of data in a report and listing the totals at the end of the result. It is used with the Totalize (TOT) run statement.

W

WFL
See work flow language.

word processing
A group of functions that process text to create documents, memos, and other kinds of reports containing textual data.

work flow language (WFL)
(1) A language used for constructing jobs that compile and run programs. WFL includes variables, expressions, and flow-of-control statements that offer the programmer a wide range of capabilities with regard to task control. (2) A language used to write jobs that control the flow of programs and tasks on the operating system.

write password
See password.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

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

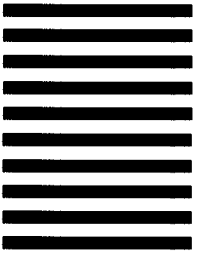
Name

Title Company

Address (Street, City, State, Zip)

Telephone Number

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

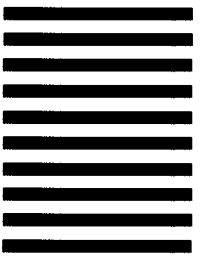


BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 1145 ST. PAUL MN 55164

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


78319274-000