SPERRY

PUBLICATIONS UPDATE

Operating System/3 (OS/3)

MAPPER® 80 Run Functions

User Guide

UP-9734–A

This Library Memo announces the release and availability of "Updating Package A to SPERRY Operating System/3 (OS/3) MAPPER 80 Run Functions User Guide", UP-9734.

MAPPER 80 is a general online report processing system that uses a report-structured data base. You do not have to understand programming to use MAPPER 80. Users can create and delete reports; manipulate data within reports; design new formats and applications; execute runs; and design new runs.

Runs are an efficient and convenient way of executing sets of MAPPER 80 run functions and are especially useful in repetitive processing.

This manual describes the MAPPER 80 run functions and provides examples of their use. It specifically describes the formats and guidelines for the run functions.

This update provides expanded coverage or:

- Data entry using upper or lowercase characters
- Debugging a run
- MAPPER 80 reserved words
- The LCH (Locate and Change) and SFS (Display Screen Format) run functions

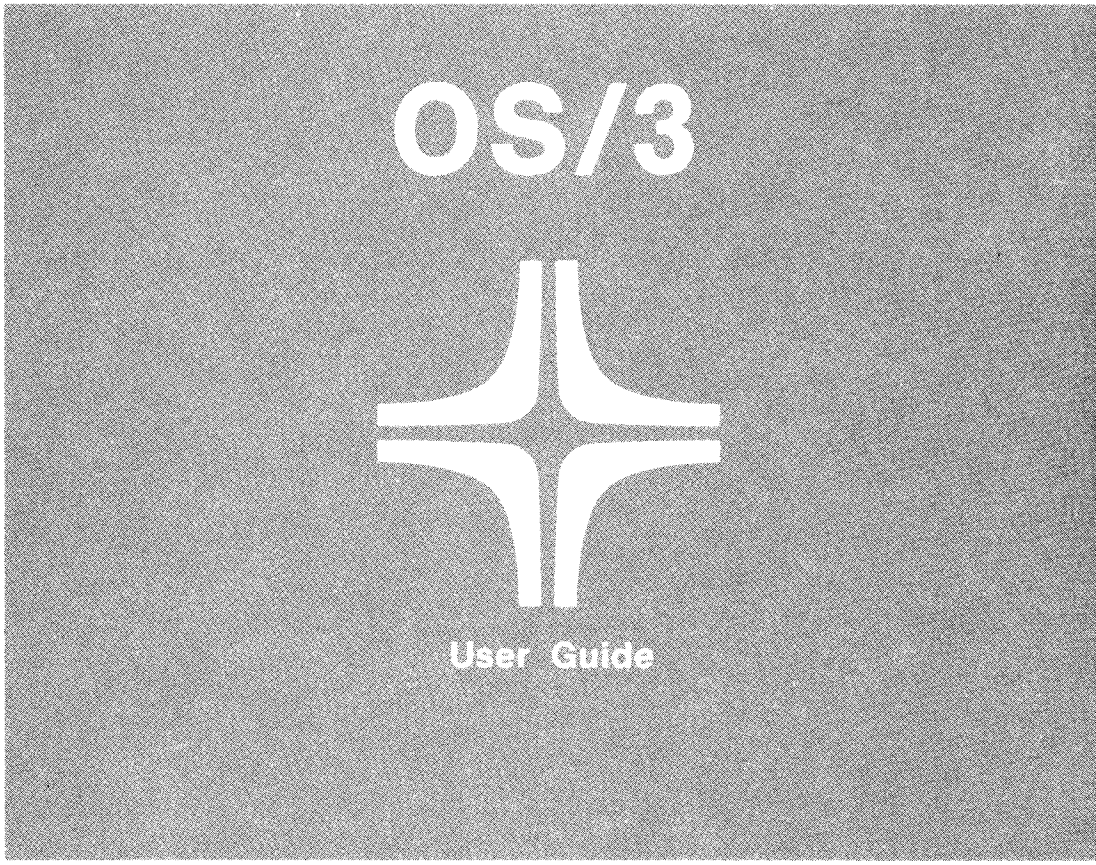
This update provides expanded coverage for:

- Data entry using upper or lowercase characters
- Debugging a run
- MAPPER 80 reserved words
- The LCH (Locate and Change) and SFS (Displays Screen Format) run functions.

Copies of Updating Package A are now available for requisitioning. Either the updating package only or the complete manual with the updating package may be requisitioned by your local Sperry representative. To receive only the updating package, order UP-9734–A. To receive the complete manual, order UP-9734.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ ,MZ, 28U, 29U,	Mailing Lists B00 and B36 (Package A to UP-9734, Cover and 17 pages plus Memo)	Library Memo for UP-9734–A
		RELEASE DATE: September, 1984

MAPPER 80 Run Functions



SPERRY

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry representative.

Sperry reserves the right to modify or revise the content of this document. No contractual obligation by Sperry regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: Update A – UP-9734
RELEASE LEVEL: 8.1 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		A
PSS	1	A
Preface	1	A
	2	Orig.
Contents	1 thru 5	Orig.
1	1 thru 7	Orig.
	8	A
	9	Orig.
2	1	A
	2 thru 11	Orig.
	12	A
	13 thru 16	Orig.
3	1 thru 4	Orig.
4	1 thru 10	Orig.
5	1 thru 5	Orig.
	6	A
	7 thru 23	Orig.
	24	A
	25 thru 40	Orig.
	41	A
	42 thru 54	Orig.
6	1 thru 5	Orig.
Appendix A	1 thru 3	Orig.
Appendix B	1, 2	Orig.
Appendix C	1, 2	Orig.
Index	1 thru 4	Orig.
User Comment Sheet		

Part/Section	Page Number	Update Level

Part/Section	Page Number	Update Level

All the technical changes are denoted by an arrow (⟹) in the margin. A downward pointing arrow (⇓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (⇑) is found. A horizontal arrow (⟹) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Preface

This manual is one in a series designed to instruct and guide the SPERRY MAPPER 80 user. It specifically describes the formats and guidelines for MAPPER 80 run functions. Runs are an efficient and convenient way of executing sets of MAPPER 80 run functions and are especially useful in repetitive processing.

A prerequisite for this manual is the manual functions user guide, UP-9735 (current version).

The MAPPER 80 library also includes current versions of the following manuals:

- MAPPER 80 manual functions user guide, UP-9735

  Describes all the manual functions and how they manipulate reports or data.

- MAPPER 80 forms generation and utilities user guide, UP-9736

  Describes the MAPLOD, MAPLST, and MAPDMS utilities.

- MAPPER 80 system operator and coordinator guide, UP-9737

  Discusses MAPPER 80 operations, including system generation and data base system coordination.

This manual is divided into six sections and three appendixes:

- Section 1. INTRODUCTION TO RUN FUNCTIONS

  Provides an overview of what is required for run creation, debugging, registration, and execution using the MAPPER 80 run functions.

- Section 2. CONVENTIONS AND RUN DESIGN CONSIDERATIONS

  Describes the format, syntax, and parameters making up the run functions.

- Section 3. PROCESSING REPORTS AND RESULTS

  Describes the four data types that are processed by the run functions.

- Section 4. WORKSTATION OPERATION

  Shows the various methods for starting and executing the run created and registered in the system.

- Section 5. MAPPER 80 RUN FUNCTIONS

  Alphabetically lists the run functions and describes their use and format. Examples are also provided.

- Section 6. RUN CREATION EXAMPLE

  Gives a walk-through of a run using the MAPPER 80 run functions.

- Appendix A. RUN FUNCTION SUMMARY

  Classifies the various run functions in a run function summary.

- Appendix B. OPTIONS TABLE

  Lists the options of the run function's option parameter.

- Appendix C. CORRESPONDENCE BETWEEN RUN FUNCTIONS AND MANUAL FUNCTIONS

  Shows the correspondence between the run functions and the manual functions.

Contents

PAGE STATUS SUMMARY

PREFACE

CONTENTS

1. INTRODUCTION

2. CONVENTIONS AND RUN DESIGN CONSIDERATIONS

5. MAPPER 80 RUN FUNCTION COMMANDS

6. RUN CREATION EXAMPLE

APPENDIXES

A. RUN FUNCTION SUMMARY

B. OPTIONS TABLE

C. CORRESPONDENCE BETWEEN RUN FUNCTIONS AND MANUAL FUNCTIONS

INDEX

USER COMMENT SHEET

FIGURES

TABLES

1. Introduction

1.1. GENERAL

A run is a MAPPER 80 report consisting of a set of run functions that specify step-by-step instructions for generating reports or results. Runs are an efficient and convenient way of executing sets of MAPPER 80 functions and are especially useful for repetitive processing, since they provide both report-generating as well as automatic data base updating capabilities. In run design, you have complete formatting flexibility.

You can intersperse the selected functional run sequence with logical decisions (e.g., branching) made on results produced or on variables accumulated in main storage. You can design runs for use in tutorial mode, thus allowing sequential interruption for display and interactive control.

You can specify most manual functions in a run report. A run report contains the specific run functions for a given run. Additional run functions with no manual function counterpart are available. Refer to Section 6 for examples of run reports and to the manual functions user guide, UP-9735, for further information on manual functions.

Figure 1–1 is a configuration showing run processing versus manual processing. Both methods process the same input, execute the equivalent functions, and produce the same output. However, run processing is more efficient.

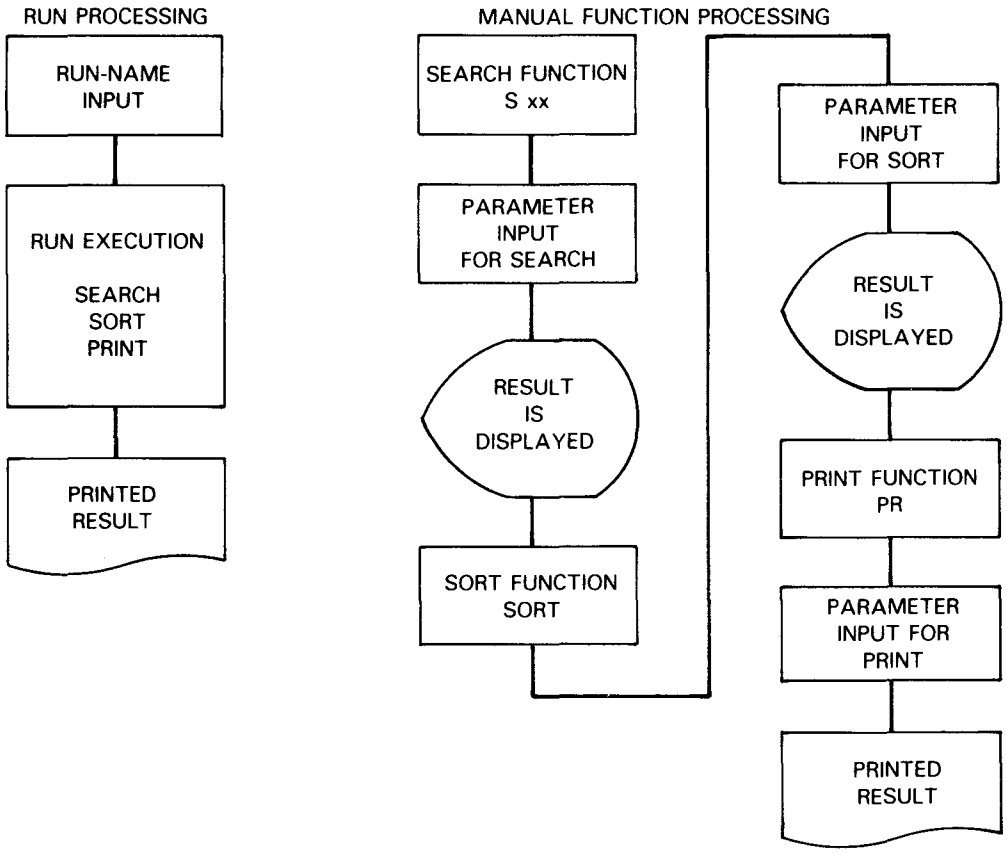


Figure 1–1. Run versus Manual Processing

1.2. CREATION, REGISTRATION, AND EXECUTION OF A RUN

To use a run function, you must carry out registration, creation, and execution of work in that order. Figure 1–2 shows these operations.

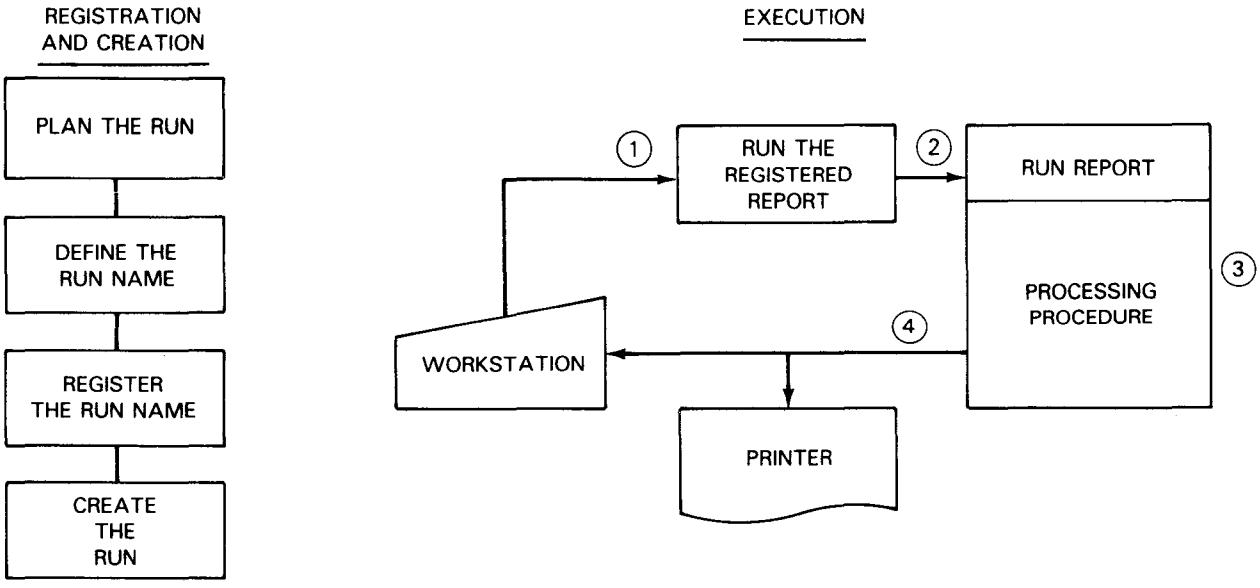


Figure 1–2. Creation, Registration, and Run Execution (Part 1 of 2)

LEGEND:

(1) Starts the run by specifying the run name via the workstation

(2) MAPPER 80 software checks whether the run name is registered.

(3) If registered, the run is processed according to the corresponding run report.

(4) Processing results are output to the workstation or printer according to run functions.

Figure 1–2. Creation, Registration, and Run Execution (Part 2 of 2)

1.3. COMPARING RUN FUNCTIONS AND MANUAL FUNCTIONS

The MAPPER 80 software enables efficient processing of runs containing several complex functions. However, there are fundamental differences between executing run functions and manual functions.

Runs that contain several functions may put a severe load on the system. In a run, where function execution is performed rapidly, virtually no pauses occur between each of the run functions. Thus, your response time may be affected. With manual updating, natural pauses between transmissions disperse the processing load.

Refer to Appendix C for the run functons and their counterpart manual functions.

1.4. CREATION AND RUN REGISTRATION

1.4.1. Run Creation Procedures

The person creating and using run functions must:

1. Create a run plan specifying the object of your run, the amount of data to be processed, and the run functions and processing outline.

2. Submit the plan to the MAPPER 80 coordinator.

3. The MAPPER 80 coordinator evaluates the plan and, if acceptable, registers the run report (RID number) and run name in the system and permits the user to create the run.

   On the other hand, if the possibility exists that the run may affect the system in a negative way, the user may have to reassess the plan. There are also times when conditional permission may be given such as specifying the time of day when the run can be executed.

4. Create the run and carry out debugging.

5. After debugging is completed and the run is ready, evaluate the run with the coordinator.

6. If the evaluation results are not satisfactory, reevaluations occur after analysis and revisions. If the results are satisfactory, register the run for use. Figure 1–3 provides a flowchart of the run creation and registration procedure.

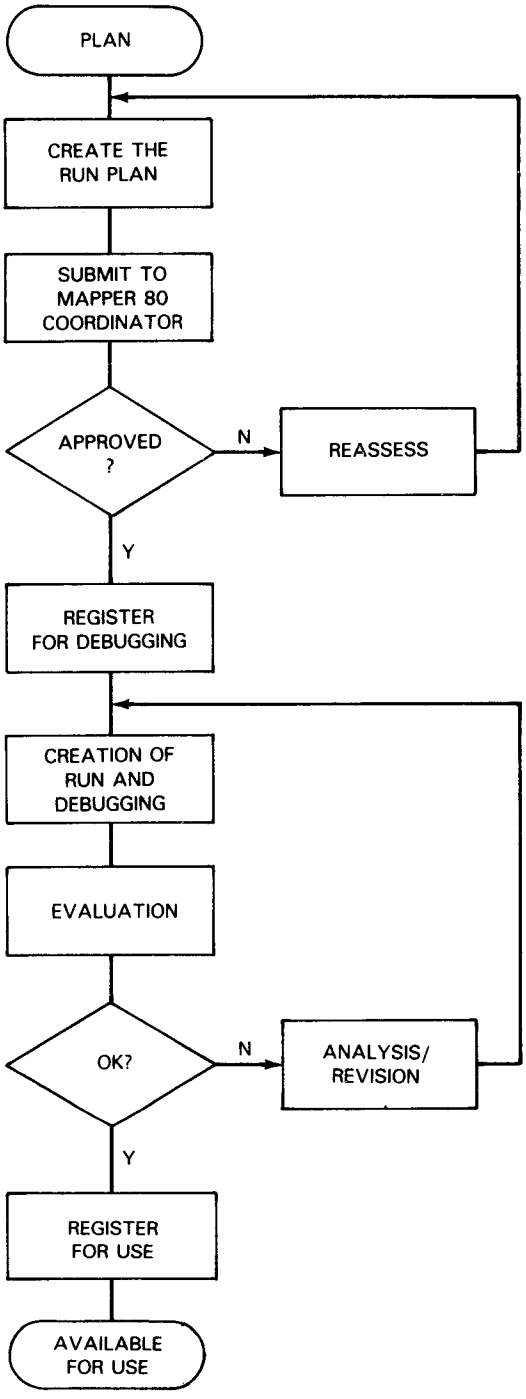


Figure 1–3. Run Creation/Registration Flowchart

1.4.2. Run Registration Guidelines

The run is registered after coordinator approval. Run registration includes the run name, mode, type, and report identity (RID) of the run being entered, as well as security and workstation identity number.

1.4.2.1. Run Name

The run name is the identifier used when executing a run. The name must be 12 or fewer characters. The first character must be alphabetic. The name cannot be one of the MAPPER 80 reserved words or a name that is already registered in the same department (e.g., the department number used when signing on).

1.4.2.2. Mode, Type, and RID

The mode, type, and RID (report identification) identify the specific report you want to access in your run. You can specify a maximum of five modes (current mode of the run, plus four additional modes) that a run can access.

1.4.2.3. Security Specifications

Security requires that you:

- Include the user identifier only when limiting the execution of a run to a specific user identifier
- Include the workstation number only when limiting execution of a run to a specific workstation

1.4.3. Support Function Considerations in Designing a Run

When designing the run, you need to know the field number, first column number, number of columns in each field, and the line length in the specified report. You can use the manual functions FLD (1.4.3.1) and CC (1.4.3.2) to display these items.

1.4.3.1. Using the FLD Manual Function to Display Report Fields

The FLD manual function displays the field number of each field of a report. To use this function, display the report you want on the screen. Key in FLD on line 0 as the function name and press the transmit key. If you input this function name when there is a hold line on the screen, an error message results.

line 0

```
LINE> fld   FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>           >
.DATE  83/07/05  09:29:51  TYPE=D  RID=001  83/06/29  JDOE        <   20 LINES>
. <<<  CORPORATE ORDER STATUS  >>>
*ST.ORDER . PRODUCT .ODR.CUST.  UNIT  .EXTENDED.REQ'D .SALE.              .
*CD.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REP .    CUSTOMER    .
*==.======.========.===.====.=======.=======.======.====.================.==
 OR 99951S GREENBOX9   2 AMCO                  750312 DJR  AMERIAN OIL CO.   73
 OR 99951S GREENBOX7   1 AMCO                  750312 DJR  AMERIAN OIL CO.   73
 OR 99951S BLACKBOX9   1 AMCO                  750312 DJR  AMERIAN OIL CO.   73
```

When FLD executes, the report displayed after line 0 moves down (rolls down) three lines and the field number of each field is indicated in the space between lines 1 and 3. The three lines that display the field numbers are then automatically held.

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN> 3  PSWD>           >
*FIELD NUMBER
                                                          1                   1
 1  2      3         4   5    6        7        8      9   0                   1
.DATE  83/07/05  09:29:51  TYPE=D  RID=001  83/06/29  JDOE        <   20 LINES>
. <<<  CORPORATE ORDER STATUS  >>>
*ST.ORDER . PRODUCT .ODR.CUST.  UNIT  .EXTENDED.REQ'D .SALE.              .
*CD.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REP .    CUSTOMER    .
*==.======.========.===.====.=======.=======.======.====.================.==
 OR 99951S GREENBOX9   2 AMCO                  750312 DJR  AMERIAN OIL CO.   73
 OR 99951S GREENBOX7   1 AMCO                  750312 DJR  AMERIAN OIL CO.   73
 OR 99951S BLACKBOX9   1 AMCO                  750312 DJR  AMERIAN OIL CO.   73
 OR 96652S GREENBOX4   2 ARCO                  750412 LSJ  ARGENTINE CORP.   23
 OR 96652S BLACKBOX5   1 ARCO                  750412 LSJ  ARGENTINE CORP.   23
 OR 96652S BLACKBOX4   1 ARCO                  750412 LSJ  ARGENTINE CORP.   23
 OR 99753S GREENBOX5   1 DICO                  750312 LSJ  DIGITAL CORP      17
```

To view the horizontal portion of the report not displayed on the screen, move the cursor to the SHFT field on line 0 and key in the number of columns to shift, then press the transmit key.

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN> 3  PSWD>           >
*
  1              1             1     1     1
  1              2             3     4     5
.DATE  83/07/05  09:29:51  TYPE=D  RID=001  83/06/29  JDOE        <   20 LINES>
. <<<  CORPORATE ORDER STATUS  >>>
*.             .             .     .     .
*.   ADDRESS   .      CITY   .STATE. ZIP .REMARK.
*.=============.=============.=====.=====.======.
  7300 CENTRAL AV NEW ORLEANS       LA 64301
  7300 CENTRAL AV NEW ORLEANS       LA 64301
  7300 CENTRAL AV NEW ORLEANS       LA 64301
  2300 5TH AVE    NEW YORK          NY 33021
  2300 5TH AVE    NEW YORK          NY 33021
  2300 5TH AVE    NEW YORK          NY 33021
  1782 NORTH ST   NEW YORK          NY 54002
  1566 COLUMBIA   WASHINGTON        DC 20001
```

1.4.3.2. Using the CC Manual Function to Display Report Column Numbers

The CC manual function displays the column numbers of the report. To use this function, display the report you want on the screen, key in CC on line 0, and press the transmit key.

An error results if you input this function when there is a hold line on the screen.

```
LINE> cc    FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>          >
.DATE  83/07/05  09:29:51  TYPE=D  RID=001  83/06/29  JDOE        <   20 LINES>
. <<<  CORPORATE ORDER STATUS  >>>
*ST.ORDER . PRODUCT .ODR.CUST.  UNIT  .EXTENDED.REQ'D .SALE.                .
*CD.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REP .    CUSTOMER     .
*==.======.=========.===.====.========.========.======.====.================.==
 OR 99951S GREENBOX9   2 AMCO                   750312 DJR  AMERIAN OIL CO.   73
 OR 99951S GREENBOX7   1 AMCO                   750312 DJR  AMERIAN OIL CO.   73
 OR 99951S BLACKBOX9   1 AMCO                   750312 DJR  AMERIAN OIL CO.   73
```

When CC executes, the report displayed after line 0 moves down three lines and the column number is displayed in the space between lines 1 and 3. The three lines that display the column number are automatically held.

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN> 3  PSWD>          >
*COLUMN NUMBER
         1111111111222222222233333333334444444444555555555566666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890
.DATE  83/07/05  09:29:51  TYPE=D  RID=001  83/06/29  JDOE        <   20 LINES>
. <<<  CORPORATE ORDER STATUS  >>>
*ST.ORDER . PRODUCT .ODR.CUST.  UNIT  .EXTENDED.REQ'D .SALE.                .
*CD.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REP .    CUSTOMER     .
*==.======.=========.===.====.========.========.======.====.================.==
 OR 99951S GREENBOX9   2 AMCO                   750312 DJR  AMERIAN OIL CO.   73
 OR 99951S GREENBOX7   1 AMCO                   750312 DJR  AMERIAN OIL CO.   73
 OR 99951S BLACKBOX9   1 AMCO                   750312 DJR  AMERIAN OIL CO.   73
 OR 96652S GREENBOX4   2 ARCO                   750412 LSJ  ARGENTINE CORP.   23
 OR 96652S BLACKBOX5   1 ARCO                   750412 LSJ  ARGENTINE CORP.   23
 OR 96652S BLACKBOX4   1 ARCO                   750412 LSJ  ARGENTINE CORP.   23
 OR 99753S GREENBOX5   1 DICO                   750312 LSJ  DIGITAL CORP      17
```

To view portions not displayed on the screen, move the cursor to the SHFT field on line 0 and key in the number of columns to shift; then press the transmit key.

```
LINE> 1     FMT>   RL>       SHFT> ▨    HLD CHR>    HLD LN> 3  PSWD>          >
*                    1111111111111111111111111111
 78888888888999999999900000000001111111111222222222
1901234567890123456789012345678901234567890123456789
.DATE  83/07/05  09:29:51  TYPE=D  RID=001  83/06/29  JDOE        <   20 LINES>
. <<<  CORPORATE ORDER STATUS  >>>
*                .              .       .     .
*   ADDRESS      .      CITY    .STATE. ZIP .REMARK.
*==============.=============.=====.=====.======.
 7300 CENTRAL AV NEW ORLEANS       LA 64301
 7300 CENTRAL AV NEW ORLEANS       LA 64301
 7300 CENTRAL AV NEW ORLEANS       LA 64301
 2300 5TH AVE    NEW YORK          NY 33021
 2300 5TH AVE    NEW YORK          NY 33021
 2300 5TH AVE    NEW YORK          NY 33021
 1782 NORTH ST   NEW YORK          NY 54002
```

1.4.4. Guidelines for Debugging a Run

During a run, MAPPER 80 software ignores the security features used with the manual functions, such as report passwords and the user identifier when replacing the reports. Therefore, you must thoroughly debug runs that change the contents of reports prior to actual use. In these cases, you debug the run without the modification steps until the results are thoroughly checked.

Prepare a list of variables when you have many variables. This helps in avoiding duplication of definitions.

There are also times when verification is necessary for the intermediate results, the details of the variables, and the processing flow for debugging runs. Inserting the display (@DSP) run function in key places and performing step-by-step verification is effective. Also include @DSP and @BRK run functions when first debugging a new run to prevent looping.

Example 1: Verifying the intermediate results

```
@SRH,M,T,R '' X-XX X,V12
@DSP,M,T,-Ø
@RNM -1.
1 2 3 4 5 6
ABCDEF
GHIJKL
@BRK DSP,M,T,-Ø
```

where M,T,R refers to the mode, type, and RID.

Example 2: Checking the details of the variables

```
@FND,mode,type,rid '' X-XX X,XX V12I3,V13I3
'V12'=V12 'V13'=V13
@BRK DSP,mode,type,-Ø
```

NOTES:

1. *To save time when debugging, include as much test data as possible in the test report. This technique is better than performing line updates after the @DSP run function for inserting test data.*

2. *To stop the run during execution, press function key F4 (also see 4.3.2).*

3. *Do not free the workstation while a run is executing. Free should be performed only when the <IDLE> logo screen is displayed.*

2. Conventions and Run Design Considerations

2.1. GENERAL

Runs consist of run functions, variables, and data. Comments may also be included in a run. If you make a syntax error when entering these, the MAPPER 80 software displays an error message when the run is executed.

If column one contains:	Meaning
@	Indicates a run function.
:	Specifies variable initiation.
Other than @ or :	Specifies data. This line is called an output line.
@.	Indicates a comment line.

MAPPER 80 run functions and data can be entered in either uppercase or lowercase; however, MAPPER 80 software translates and stores all data in uppercase.

2.1.1. Style Conventions in This Manual

The style conventions used in this manual help make information understandable when you want to use MAPPER 80 software.

The following items appear in uppercase letters:

- MAPPER 80 run functions, preceded by an at sign (@), for example, @CHG, @TOT
- Reserved words (e.g., INPUT$)

In text, user entries also appear in uppercase letters. However, unless otherwise indicated, you can make all your entries without shifting to lowercase letters.

The format of a run control statement has these things in common:

- The call is capitalized (e.g., CHG).
- A delta sign (△), when used, indicates a space is needed.
- Fields or subfields enclosed in brackets are optional. In the following example, *field2, subfield1,* and *subfield2* are optional:

```
field1 [field2 subfield1,subfield2]
```

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

```
@AUX,2,b,2,999,cop,,,y,,2
```

- Braces around an item mean that you may choose from among the items listed. For example:

$\left\{\begin{matrix}item1\\item2\end{matrix}\right\}$ or {item1 | item2}

2.1.2. Run Function Format

Every run function begins with the at sign (@) in the first column; however, within the run you can have @. comment lines, variables, or output data also starting in column 1. Table 2–1 alphabetically lists the run functions.

Table 2–1. Run Functions

@ADD	(APPEND REPORT)
@ADR	(ADD REPORT)
@AUX	(AUXILIARY DEVICE)
@BFN	(BINARY FIND)
@BRK	(BREAK)
@CHG	(CHANGE)
@COP	(COP DEVICE)
@DEL	(DELETE LINES)
@DLR	(DELETE REPORT)
@DSP	(DISPLAY REPORT)
@DUP	(DUPLICATE REPORT)
@ESR	(EXIT SUBROUTINE)
@FND	(FIND)
@GTO	(BRANCH)
@IF	(CONDITIONAL)
@IND	(INDEX)
@INS	(INSERT DATA)
@LCH	(LOCATE AND CHANGE)
@LLN	(LAST LINE NUMBER)
@LNX	(DUPLICATE LINE)
@LN+	(ADD LINE)
@LN–	(DELETE LINE)

@LOK	(UPDATE LOCK)
@MAU	(MATCH UPDATE)
@MCH	(MATCH)
@PRT	(PRINT)
@RDC	(READ CONTINUOUS)
@RDL	(READ LINE)
@REL	(RELEASE)
@REP	(REPLACE REPORT)
@RLN	(READ LINE NEXT)
@RNM	(RENAME)
@RSR	(RUN SUBROUTINE)
@RUN	(START RUN)
@SFS	(DISPLAY SCREEN FORMAT)
@SOR	(SORT)
@SRH	(SEARCH)
@SRU	(SEARCH UPDATE)
@SUB	(SUBTOTALLING)
@TOT	(TOTALIZE)
@ULK	(UNLOCK)
@UPD	(UPDATE)
@WRL	(WRITE LINE)
@XIT	(SIGN OFF)

NOTE:

These run functions are categorized and defined in Appendix A.

There is no single general format that can apply to all the run functions. The specific format of each run function is given in Section 5. However, all the run functions contain several or all the elements listed in this format example:

```
@label:function,mode,type,rid options characters parameters variables
```

where:

@
: Indicates that this is a run function.

label:
: Is a number referenced by the @GTO (branch) run function to change the flow of control during execution of a run. Use decimal numbers within the range 1 through 199. Label numbers must be unique within the run.

Example:

```
@IF V1 = V2 GTO 100;GTO 101
            .
            .
            .
@100:CHG V1 0
            .
            .
            .
@101:CHG V1 V1 + 1
```

function
: Specifies a run function and is three alphabetic characters long (except @IF).

mode
: Indicates the decimal mode number (0 through 999). Specifies read only access or read and write access by an odd- or even-number configuration.

type
: Is specified by the letters A through I.

rid
: Indicates the identification number of the report (0 through 999).

options
: Alphabetic characters that indicate choices. If you don't specify options, you must use a double apostrophe (''). Enclose options that require special characters in apostrophes. Refer to Appendix B for the various options available and their descriptions.

Example:

`D` Specifies option D.

`DH` Specifies options D and H.

`''` Specifies no options (two apostrophes).

`'D/'` Specifies the D and slash options.

`characters`

Specifies column number in a line or field, number of characters, or an entire field. When specifying multiple fields, separate each with a comma. Define the character field in the following three formats:

`Fn` Specifies the field of field number n. The n is a decimal number beginning with 1.

`Fn-l` Specifies l characters from the first column field number n. The l is a decimal number beginning with 1.

`m-l` Specifies l characters from column number m. The m is a decimal number beginning with 1. Unlike the other two formats, column numbers can be specified freely regardless of the field number.

Examples:

`F4` Specifies the entire field of field number 4.

`F4-4` Specifies four characters from the first column of field number 4.

`F3,F6-6` First field: Entire field of field number 3.
Second field: Six characters from the first column of field number 6.

`2-6` Six characters from the second column.

`2-6,9-10` First field: Six characters from the second column.
Second field: 10 characters from the ninth column.

`parameters`

Defines the line type to be processed. It also defines the variables, constants, and expressions in relation to the character field that carry out the run function.

Example:

```
@SRH,500,B '' F4 *,CAT V1,V2
```

`@SRH` Search run function is indicated for all RIDs in type B, mode 500.

`*,CAT` This is a parameter with an asterisk (*) line-type, searching for the three characters CAT in field 4 (F4).

`variables`
Describes the name of a variable, in this case V1 and V2, in which processing results and the operation results are entered.

Example 1:

Referencing one report in which you want to perform a range search:

```
@5:SRH,102,B,1 D F2 *,0/R,9 V4I5,V5I5
```

where:

`@`	Indicates a run function.
`5:`	Is the label.
`SRH`	Specifies the particular run function.
`102`	Is the mode (filing cabinet concept).
`B`	Is the type (drawer of the filing cabinet concept).
`1`	Is the RID (specific report number in drawer concept).
`D`	Specifies the D option. See Appendix B.
`F2`	Specifies the characters.
`*,0/R,9`	Specifies the parameters.
`V4I5,V5I5`	Specifies the variables.

Example 2:

Referencing two reports:

```
@6:MCH,102,B,1,102,D,30 ''  F3,F4 *,1,A F1,F3 *,1,A
```

where:

`@`	Indicates a run function.
`6:`	Is the label.

MCH	Specifies the particular run function.
102,B,1	Specifies mode, type, and RID of the issuing report.
102,D,30	Specifies mode, type, and RID of the receiving report.
''	Specifies no options.
F3,F4	Specifies characters of the issuing report.
*,1,A	Specifies parameters of the issuing report.
F1,F3	Specifies characters of the receiving report.
*,1,A	Specifies parameters of the receiving report.

Example 3:

Run function not referencing report:

```
@8:CHG V1 V2 + V3
```

V1 is the sum of variables V2 and V3.

Example 4:

Run function not referencing report:

```
@RUN RUN2,V1,V2,SAM
```

The current run executes run 2 and passes to run 2 data contained in variables V1 and V2 and the constant SAM.

2.1.3. Variable Format

Variables always begin with the letter V followed by a number from 1 to 199. They are work buffers. You can define the variable format type, number of columns, and initial value at the same time. Variables can be defined and redefined anywhere within the run.

The format for variable definition is:

```
:Vnnn type size [constant
```

where:

:	Indicates variable definition.
Vnnn	Describes the variable name (V1-V199).

`type` Specifies the variable types (A,H,F,I,S). Refer to 2.1.2.1.

`size` Indicates the number of columns.

`[(tab)` Specifies the tab delimiter ([) between the variables and constants.

`constant` Specifies constants or character strings as the initial value you want to store in the variable.

Example:

`:V102H12[123456789012` Defines type H, number of columns (12), and the initial value of: (123456789012) of variable V102.

2.1.3.1. Variable Types

The following five types of variables are available:

- Type A

  Maximum of 16 alphanumeric and special characters. You can perform arithmetic operations with type A variables via the @CHG run function.

Example:

`:V1A10[ABCDE12345` (Variable 1 with a size of 10 alphanumeric characters)

- Type F

  Real numbers to a maximum of 16 characters, of which 14 may be the fractional portion. Use type F for arithmetic operations that contain fractional numbers. These variables must contain decimal points. Both sign and decimal point are significant digits; include them in variable size. Positive numbers do not require a sign; negative numbers require a minus (–) sign.

  Examples:

  `:V1F6.2[123.43` (Variable 1 is a positive number with a total size of 6 and two decimal positions.)

  `:V2F7.2[-123.43` (Variable 2 is a negative number.)

- Type H

  Maximum of 16 alphanumeric and special characters. You cannot use type H for arithmetic operations. Use it to enter character strings.

Example:

```
:V3H16[MAPPER80.REPORTS
```
(Variable 3 with a size of 16 alphanumeric characters)

- Type I

  Integers to a maximum of 16 characters. Use type I for arithmetic operations. The results are entered in an integer format. The sign is a significant character and must be included in the variable size. Positive numbers do not require a sign; however, negative numbers require a minus (–) sign.

  Example:

```
:V57I6[12345
```
(Variable 57 with a size of 6 specifying a positive number)

```
:V58I6[-12345
```
(Variable 58 with a size of 6 specifying a negative number)

- Type S

  Alphanumeric and special characters with a maximum of 132 characters. Type S cannot be used for arithmetic operations. Use them to make character strings with a maximum of 132 characters.

  Example:

```
:V152S30[ABCDEFGHIJKLMNOPQRSTUVWXYZ1234
```
(Variable 152 with a size of 30 characters)

2.1.3.2. Methods to Initialize and Define Variables

Three methods can be used for initializing and defining variables. These include the:

- Colon (:)
- Change (@CHG) run function
- Run function

2.1.3.2.1. Using the Colon Method to Initialize and Define Variables

Variables can be initialized by the colon method. To use the colon method, enter a colon in column 1 of the run, a variable name Vnnn tsss starting in column 2, a tab character ([), and the initial value of the variable. When defining multiple variables, delimit the variables with spaces and semicolons.

In the variable name Vnnn tsss,

Vnnn Specifies V1 to V199.

t Specifies the type (A,F,H,I,S).

sss Specifies the number of characters.

Example:

```
:V1A3[ABC
:V1A3[ABC ;V2H3[DEF ;V3I3[123 ;V4F5.2[12.34
```

2.1.3.2.2. Using the @CHG Method to Initialize and Define Variables

To use the change method, enter an @CHG run function, followed by one or more spaces. Next, define the variable, followed by one or more spaces. Then, define the value of the variable. You can include multiple @CHG run functions on the same line.

Example:

```
@CHG V1A3 ABC
@CHG V2I3 123 CHG V3F5.2 12.34
```

2.1.3.2.3. Using the Run Function Methods to Initialize and Define Variables

Several run functions initialize variables: (@RDC, @RDL, and @RLN). The @FND run function places certain information into variables, and the @TOT run function puts values into variables.

Example:

```
@RDL,2,B,6,8 F1,F2 V1ØA2,V11A6
```

Reads in the first two characters from the first field (F1) of line 8 of the report (2, B, 6). Reads in six characters from the second field (F2) and enters them in the type A variables V10 and V11.

2.1.3.3. Variable Initialization and Manipulation Techniques

Entering blanks in the variables involves enclosing one blank character in apostrophes or writing blank characters equal to the size of the variable.

Example:

```
:V1A4['Δ'
:V1A4[ΔΔΔΔ
@CHG V1A4 'Δ'
```

It is also possible to reference and process parts of character strings in a variable.

Example:

```
Vn(m-l)
```

Vn is the variable name between V1 and V199; m is the position of the character in the variable (1 through 132); l is the number of characters (1 through 132).

The variables m and l can be of the A or I type.

The value of the variable must be from 1 through 132.

Example:

```
:V1H6[YYMMDD
```

`@CHG V2H2 V1(3-2)`	MM (month) enters V2.
`@CHG V3I1 3 CHG V4I1 2`	V3 assigned a value of 3, and V4 assigned a value of 2
`@CHG V5H2 V1(V3-V4)`	MM enters V5.

It is also possible to modify and initialize variables. This is accomplished by using the @CHG run function and @INS run function.

Example:

`@CHG V10A3 AAA`	Defines V10 contents as AAA.
`@CHG V10 BBB`	Defines V10 contents as BBB.
`@CHG V11I3 123`	Defines contents of V11 as 123.
`@CHG V11 V11+1`	The contents of V11 becomes 123 + 1 = 124.
`:V12S15[0123456789ABCDE`	Defines contents of V12 (initial value).
`@INS V10 V12(1-3)`	The contents of V12 becomes BBB3456789ABCDE.
`@INS V11(2-2) V12(14-2)`	The contents of V12 becomes BBB3456789ABC24.
`@CHG V13I4 4`	Defines contents of V13 as 4.
`@CHG V10 V12 (V13-3)`	Contents of V10 becomes 345.

2.1.4. Data (Output Line) Format

When the first column of the run function starts with a character other than @ or :, it is considered an output line. These lines are written sequentially in the output area. Refer also to 3.6. Variables described in the output line are written to the output area after being replaced by substitute values.

Example:

```
:V1A5[ABCDE;V2I5[12345      Defined variable line
      .
      .
      .
OUTPUT DATA V1,V2           Output line

OUTPUT DATA ABCDE,12345     Data is substituted into variables V1 and V2 and then
                            written to the output area.
```

2.1.5. Comment Format

Comments may be entered freely in the run. Comments are useful for describing the input to the run, the logic flow of the run, and the output from the run. Comments may include any data that the user feels is pertinent.

Comments do not appear in the results produced by the run, nor do they affect the processing performed by the run. There are two comment formats: comment lines and commenting after a run function.

2.1.5.1. Comment Lines

When a period immediately follows an @, that line is considered a comment line.

Example:

```
@. THIS IS A COMMENT LINE
```

When a period immediately follows a label, that line is considered a comment line. This type of line may be the destination line of an @GTO run function.

Example:

```
@GTO 5Ø.       THIS IS A COMMENT LINE
  .
  .
  .
@5Ø.           THIS IS A COMMENT LINE
```

Shows branching to the label 50 line.

2.1.5.2. Commenting after a Run Function

To enter comments on a run function line, terminate the run statement with the character string space period space (△.△).

Example:

```
@CHG V1A6 DATE1$ . INITIALIZE V1 with DATE
```

2.2. FIELDS AND SUBFIELDS

Run function statements consist of fields and subfields. Fields are delimited by spaces. Each field can also be delimited into subfields with commas, slashes (/), and hyphens.

Example:

		@5:SRH,102,B,1	DH	F1-4	[,A102/R,B200	V415,V516
Field Number		1	2	3	4	5
Subfield	1	@5:SRH	DH	F1	[(Tab)	V415
	2	102		4	A102	V516
	3	B			R	
	4	1			B200	

2.3. RESERVED WORDS

Reserved words end with a dollar sign ($) and contain specific data. For example, the reserved word TIME$ contains the current time in the format HH:MM:SS. Only MAPPER 80 software can initialize the reserved words. Access to the reserved words is limited to the @CHG run function. The user can initialize a variable with the value of the reserved word by the @CHG function. The initialized variable can then be used by other run functions.

Examples:

`@CHG V1A8 TIME$`	Enter time in the variable.
`TIME=V1`	Writes time in output area.

NOTE:

When creating MAPPER 80 runs, character strings ending in a dollar sign ($) should be enclosed in apostrophes. This prevents character strings from being mistaken for reserved words.

Example:

```
'STOP$'
```

The reserved words used by the MAPPER 80 system are listed in Table 2–2.

Table 2–2. MAPPER 80 Reserved Words

Reserved Word	Meaning
CHARS$	Specifies the maximum number of characters per line for the current result.
DATE1$	Specifies the current date. The format is yymmdd, where: yy is the year; mm is the month; and dd is the day.
DEPN$	Specifies the department number of the user when signing on.
DLINE$	Specifies the line number of the first line displayed.
FMT$	Specifies the format number of the displayed report.
INPUT$	Specifies external data input to the run. The maximum number of variables is 40 with a total of 320 characters.
LINE$	Specifies line number to be read with the next @RLN run function.
LLP$	Specifies the number of run functions that are processed.
MODE$	Specifies mode number of the current result.
MODE1$	Specifies the new mode number of the mode referenced from the workstation executing the run.
OLINE$	Specifies the line number after the line currently being written in the output area.
STAT1	Specifies status 1 of the special function. This includes status or number. Initialize with I type variable.
STAT2	Specifies status 2 of the special function. This includes status or number. Initialize with I type variable.
STNUM$	Specifies the workstation identity requesting run functions.
TIME$	Specifies the current time. The format is hh:mm:ss, where: hh is hours; mm is minutes; and ss is seconds.
TYPE$	Specifies the type of the current result.
USER$	Specifies the user identifier (identifier used when signing on).

2.4. GUIDELINES FOR CONFIGURING THE RUN FUNCTION

Consider the following guidelines when configuring the run function:

1. Define multiple run functions on the same line by placing one or more blanks between each run function.

2. The semicolon can fulfill two roles:

   - When defining two or more variables, use a semicolon between each variable.

     Example:

     ```
     :V1A3[ABC;V2I4[1234
     ```

   - When you use the @IF run function, a semicolon delimits multiple conditions selected in the same line.

     Example:

     ```
     @IF V1 = 3 GTO 9;IF V1 < 6 GTO 8;IF V1 = 6 GTO 1Ø
     @GTO END
     ```

     If the value V1 is equal to 3, go to label 9.
     If the value of V1 is not equal to 3 and is less than 6, go to label 8.
     If the value of V1 is equal to 6, go to label 10.
     If the value of V1 is greater than 6, the next run function is executed (in this example, the run ends).

3. When describing special characters with run functions and variables for purposes other than delimiting characters (constants or character strings), enter the character within apostrophes.

   Example:

   ```
   @CHG V1 V2 '/' V3
   @SRH,1ØØ,B,2 DH F1-1 [,'@' V1I3,V2I5
   ```

4. When you use a space character for reasons other than delimiting the field, enclose it in apostrophes.

5. You do not have to use subfields at the end of a field. Omit subfields by adding a series of commas. Fields must not end with a comma.

   Example:

1,2,3,4	No omissions
,2,3,4	First subfield omitted
1,2,,4	Third subfield omitted
1,2,3	Last subfield omitted

6. Subfields can be defined with variables.

Example:

```
:V1I3[102;V2A1[B;V3I3[1
:V6H4[A100;V7H4[B200
@5:SRH,V1,V2,V3 D F1-4 [,V6/R,V7 V4I5,V5I6
```

This is the same as:

```
@5:SRH,102,B,1 D F1-4 [,A102/R,B200 V4I5,V5I6
```

7. Although multiple run functions can usually be combined in one statement, run functions must not be defined after the following functions: @DSP, @ESR, @RDC, @RSR, @RUN, and @SUB.

8. The label must be from 1 through 199.

9. The variable name must be from V1 through V199.

10. When executing the @WRL, @LN+, @LN-, and @LNX run functions, you must lock the report being updated with the @LOK run function.

11. An output line must follow the @SUB and @RDC run functions. You can specify only one line for output.

    Example:

    ```
    @RDC,102,C,1,30,100,[,40 F1,F2,F3 V1A9,V2A5,V3I6
    V1 V2 V3
    ```

    From the report in mode 102, type C, RID 1, starting at line 30 for 100 lines, read the contents of fields F1, F2, and F3 into variables V1, V2, and V3, respectively. Then write V1, V2, and V3 to the output area. Product name equals V1, classification equals V2, cost equals V3.

    Example:

    ```
    @SUB,102,C,1 E F2,F3,F8 [,S,+,+ V1I3,V2A5,V3I8,V4I4
    V1 V2 V3 V4
    ```

    Accumulate subtotals for fields F3 and F8 into variables V3 and V4 for each change in field F2 (subkey). Include the number of entries into V1 for each change in field F2 (subkey). Then write V1, V2, V3, and V4 to the output area. Total, classification equals V2, number of entries equals V1, cost equals V3, and quantity equals V4.

12. If a number follows the letter V, it is considered a variable name. If you follow the letter V with a number that is a constant or character string, you must enclose it in apostrophes.

Example:

```
@CHG V1I3 123
THE VALUE OF 'V1' is V1
```

Variable V1 is initialized to 123. The next line is written to the output area. The value was substituted for the variable to produce the following line:

THE VALUE OF V1 IS 123.

13. The back slash (\) is used when a run statement is carried over to the next line.

    The @ sign is not specified on the continuation lines. Up to nine continuation lines may be used. The maximum number of characters in a single run command is 256. Specify line continuation in the following ways:

    - After a field: space back slash (△\)
    - After a subfield: back slash (\)

    Examples:

```
@10:MCH,102,B,2,102,D,1 '' F1,F4,F11 [,1,A,B△\
F1,F3,F8 [,1A,B

@20:WRL,102,B,2,102 F1,F2,F3,F4 [,V1,V2,V3,V4/[,V11,\
V12,V13,V14/[,V21,V22,V23,V24
```

14. All required fields, within each instruction must be defined.

3. Processing Reports and Results

3.1. GENERAL

The run functions, as compared to the manual functions, require closer attention when processing reports and results. This section describes the data types processed by run functions, including the results.

3.2. DATA TYPES

When you use manual functions, such as TOT and SORT, results are created. If you use this result in the execution of the next manual function, a new result is created and the result prior to the execution is lost. Run functions, on the contrary, allow you to save the result prior to the execution.

The four types of data that the MAPPER 80 run functions process are shown in Figure 3–1.

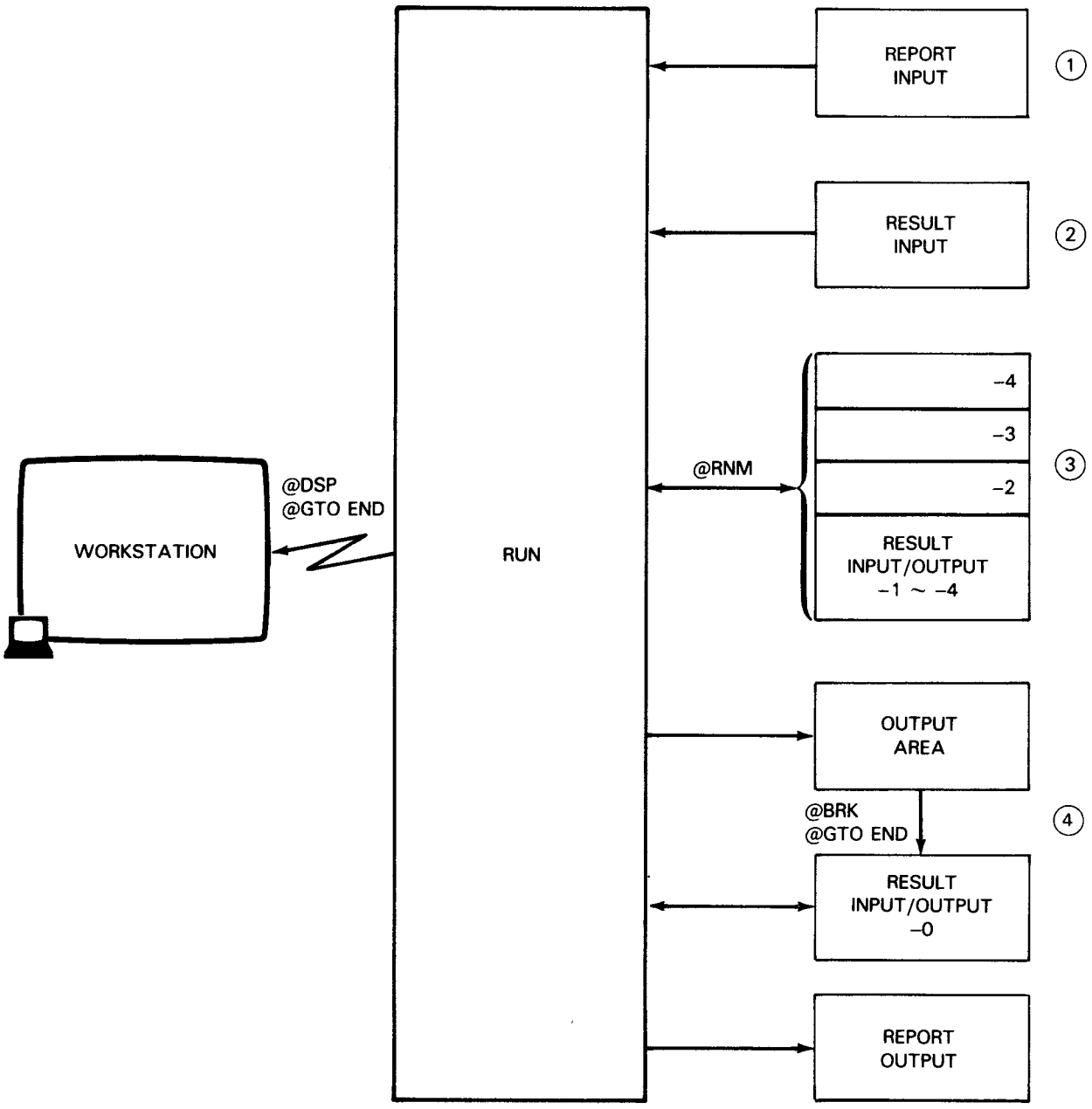
3.3. PROCESSING REGULAR REPORTS

You can process regular reports in the MAPPER 80 data base by specifying the mode, type, and report number (RID) of the report in the run function. Usually, only reports in one mode are accessible. However, with the approval of the MAPPER 80 coordinator, you can access reports in other modes.

3.4. PROCESSING RESULTS

You can use results generated by manual functions as input to a run. These manual functions can include TOT and SORT.

Results produced by the execution of a run function are temporarily stored in the current mode and type under a RID number of –0. If another run function that produces a result is executed, the previous –0 result is replaced with this result. Only one result at a time exists under the RID number –0. To save a –0 result for further processing at a later point in the run, you must rename it by using the RNM run function and then access the result under a RID number of –1 to –4.



Data Types:

(1) Regular data in the MAPPER 80 data base

(2) Results generated by the manual functions that are input to the run

(3) Results (–1 to –4) temporarily saved using the @RNM run function from the result (–0) created by the run (3.4.1)

(4) Result (–0) created by the @BRK run function are temporarily stored in the current mode, type, and RID (–0). Refer to 3.5.1.

Figure 3–1. Data Types Handled by Run Functions

3.4.1. Using the @RNM Run Function to Process Reports

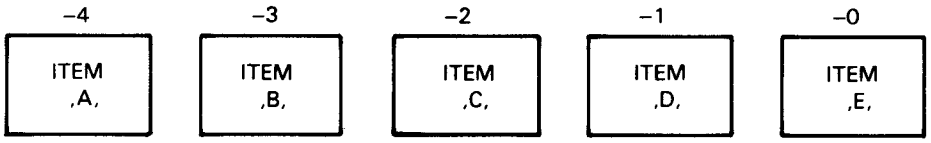
Only one result exists at any time when processing the report with the manual or run functions of MAPPER 80 software. When it is necessary to save a certain number of results during run execution, you can temporarily save these results by using the @RNM run function. This command saves the current result –0 by changing the result number to a number between –1 and –4. The maximum number of results that you can save using the @RNM function is four (–1 to –4). You access these renamed results by specifying the result number (–0 to –4) as the RID number. For example, to display the –3 result from mode 102, type B, you issue the following command:

```
@DSP,1Ø2,B,-3
```

Examples:

1. `@SRH,M,T,R '' F1 ,A V1,V2` `@RNM -4`	Search for item A in field 1 of mode (m), type (T), and RID (R) and save the result as result number –4.
2. `@SRH,M,T,R '' F1 ,B V1,V2` `@RNM -3`	Search for item B in field 1 of mode (M), type (T), and RID (R) and save the result as result number –3.
3. `@SRH,M,T,R '' F1 ,C V1,V2` `@RNM -2`	Search for item C in field 1 of mode (M), type (T), and RID (R) and save the result as result number –2.
4. `@SRH,M,T,R '' F1 ,D V1,V2` `@RNM -1`	Search for item D in field 1 of mode (M), type (T), and RID (R) and save the result as result number –1.
5. `@SRH,M,T,R '' F1 ,E V1,V2`	Search for item E in field 1 of mode (M), type (T), and RID (R). (Result automatically saved as –0)

The following five reports are created as a result of these five examples:



The results created by the searches can be printed on the auxiliary printer with the following types of run function statements.

```
@AUX,M,T,-4
@AUX,M,T,-3
@AUX,M,T,-2
@AUX,M,T,-1
@AUX,M,T,-Ø
```

3.5. OUTPUT AREA

An output area is the temporary work area used by MAPPER 80 software to store data prior to creating results.

All lines in a run report starting with characters other than @ or : are called output lines and are written sequentially in the output area. (See Figure 3–2.)

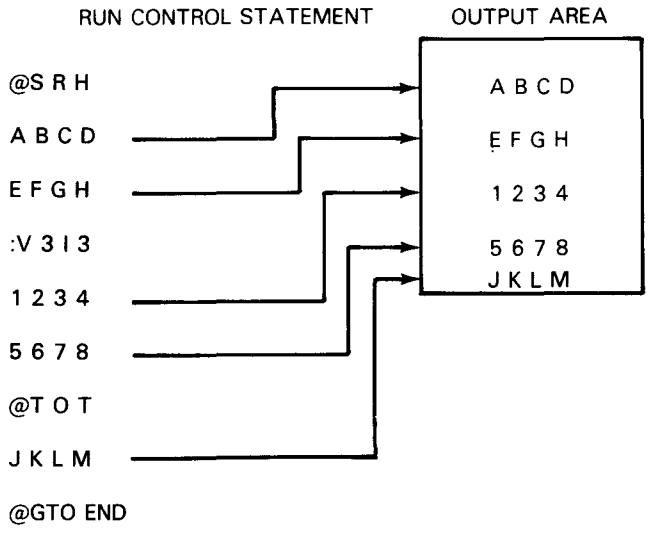


Figure 3–2. Output Area Containing Output Lines

3.5.1. Using the @BRK Run Function with the Run Output Area

When the @BRK run function executes, the contents of the run output area up to that time becomes result (–0). The contents of the output area is then cleared and subsequent output lines are written to the cleared output area starting at line 2. MAPPER 80 software uses the first line to set the date.

The run report's mode and type numbers are automatically assigned to the output area when starting the run but can be changed by including a new mode and type in the @BRK run function. The new mode and type are assigned to subsequent results created by the @BRK run function, but not to the current result. See 5.1.5 for more information.

3.5.2. Using the @GTO END Run Function with the Run Output Area

When the @GTO END function executes, the run ends, the contents of the output area become the result, and the result is displayed on the screen. Manual functions can be used to further manipulate the displayed results. See 5.1.14 for more information.

4. Workstation Operation

4.1. GENERAL

This section describes the operation of the MAPPER 80 workstation when using the run functions.

4.2. STARTING THE RUN

You can start the run by using any of the access methods found in the subsequent subsections.

4.2.1. Formal Access Method

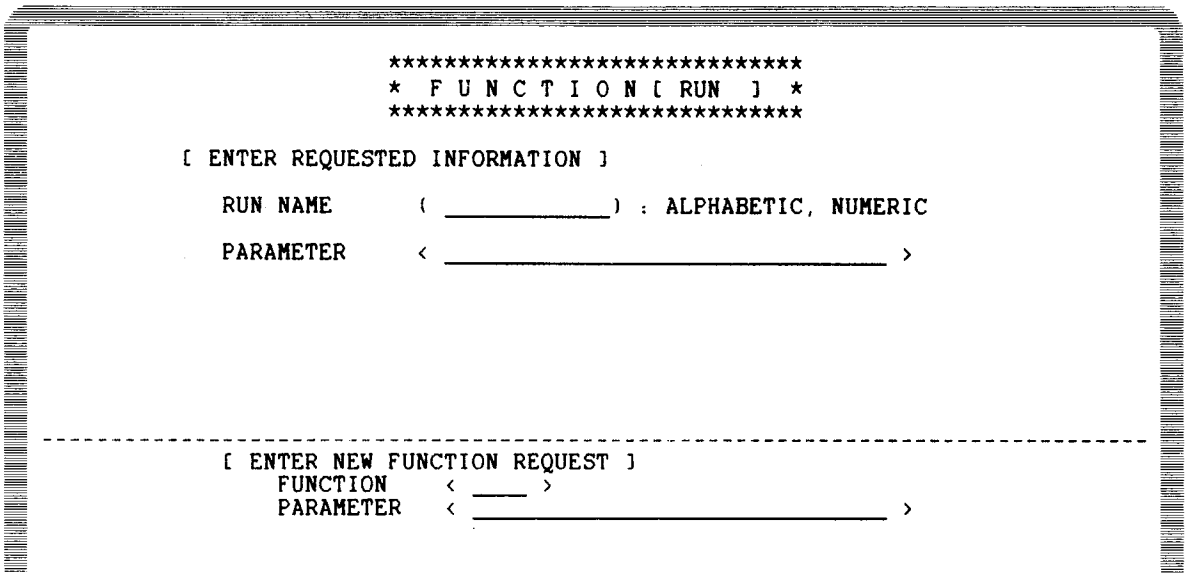
The run function name is input after the start of entry (SOE) character of line 0 or as the function name in the home position when a report is displayed. It may also be input as:

- the function name field of the user logo; or
- the function name in the exchange function name field of the function request screen.

```
                ******************************
                *  M A P P E R  8 0 ( 2.00 ) *
                *                            *
                *    USER  [ JDOE          ] *
                *    D-NO. [ 007 ]           *
                *    MODE  [ 102 ]           *
                *    WORK STATION   [ T01 ]  *
                *         83 / 07 / 07       *
                ******************************

[ ENTER FUNCTION REQUEST ]

       FUNCTION     ( run_ )
       PARAMETER    < ____________________________ >
```



NOTES:

1. *The run name field contains the run name of the run to be entered.*

2. *The parameter field contains the input parameter to be passed to the run. This is optional. The parameter consists of a single, nondelimited character string with a maximum length of 32 characters.*

3. *Run execution begins after all fields are filled in and transmitted. When you start the run with the fast access methods, the maximum length, including the run name, is 30 characters.*

The parameter, which is entered into the reserved word INPUT$, can be divided into segments by the @CHG run function contained in the called run.

```
@CHG INPUT$ V1I3,V2A1,V3I3
```

In the following examples of the fast access method, parameter 102B002 is passed to the run, JDOERUN1.

The @CHG run function in JDOERUN1 transfers data from INPUT$ to the variables based on the defined length of the variables. The length of V1 is 3, V2 is 1, and V3 is 3.

The first three characters of INPUT$ are entered into V1, the fourth character is entered into V2, and the last three characters are entered into V3.

The contents of the variables at the completion of the @CHG run function is:

```
V1=1Ø2
V2=B
V3=ØØ2
```

The variables can now be used by other run functions, such as:

```
@DSP,V1,V2,V3
```

4.2.2. Fast Access Method 1

With fast access method 1, the input begins after the SOE of line 0, at the home position, or in either the function exchange field of the function request screen or in the function field of the user logo. The format is:

```
▷ RUN run-name,[parameter]
```

Example of input from line 0:

```
LINE> run jdoerun1,102b002   SHFT>      HLD CHR>    HLD LN>    PSWD>          >
.DATE  83/07/05  10:47:50  TYPE=B  RID=002  83/06/29  JDOE        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 IP 741224 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 IP 741225 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 IP 741219 LS BLACKBOX2 637071        84353 INTR 741218 741219
```

Example of input from the function exchange fields of a function request screen:

```
                    ********************************
                    *  F U N C T I O N [ TOT  ]  *
                    ********************************

        [ ENTER REQUESTED INFORMATION ]

           REPORT NO.     ( ___ )     : '0' - '999' OR '-'

           T Y P E        < _ >       : 'A' - 'I' (WHEN REPORT NO. IS '-', YOU
                                                   CAN OMIT TYPE)
           FORMAT         < _ >       : '1' - '6'




-------------------------------------------------------------------------------
           [ ENTER NEW FUNCTION REQUEST ]
               FUNCTION     < run_ >
               PARAMETER    < jdoerun1,102b002_____________ >
```

Example of input from the user logo screen:

```
                ******************************
                *  M A P P E R  8 0 ( 2.00 ) *
                *                            *
                *    USER  [ JDOE          ] *
                *    D-NO. [ 007 ]           *
                *    MODE  [ 102 ]           *
                *    WORK STATION   [ T01 ]  *
                *         83 / 07 / 13       *
                ******************************

[ ENTER FUNCTION REQUEST ]

      FUNCTION     ( run_ )
      PARAMETER    < jdoerun1,102b002_____________ >
```

Run execution starts when you press the transmit key.

4.2.3. Fast Access Method 2

With fast access method 2, enter input after the at sign (@) is placed in front of the run name. Key in the run function after the SOE of line 0 or the home position. The format appears as:

```
▷ @run-name,[parameter]
```

```
LINE> @jdoerun1,102b002      SHFT>      HLD CHR>    HLD LN>    PSWD>           >
.DATE  83/07/05  10:47:50  TYPE=B  RID=002  83/06/29  JDOE        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 IP 741224 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 IP 741225 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 IP 741219 LS BLACKBOX2 637071        84353 INTR 741218 741219
```

Run execution starts when you press the transmit key.

4.3. RUN EXECUTION/TERMINATION VIA WORKSTATION

4.3.1. Workstation Display during Run Execution

When run execution starts, the run logo message appears on the first line of the workstation display. The rest of the lines on the screen remain in the state they were in prior to the start of the run. These lines remain on the screen until they are replaced by another display (report or result display, error message, user logo, etc) that is related to the execution of the run.

```
              MAPPER80 RUN FUNCTION PROCESSING <F4...CANCEL>

                     ******************************
                     *  F U N C T I O N [ RUN  ]  *
                     ******************************

        [ ENTER REQUESTED INFORMATION ]

           RUN NAME       ( jdoerun1____) : ALPHABETIC, NUMERIC

           PARAMETER      < 102b002_____________________________ >




-------------------------------------------------------------------------------
              [ ENTER NEW FUNCTION REQUEST ]
                 FUNCTION    < ____ >
                 PARAMETER   < ____________________________________ >
```

4.3.2. Workstation Termination

You can terminate the run, but only when the run logo is displayed, by pressing function key F4. The following message appears on the bottom line of the screen. If you enter a Y response, the run terminates.

```
DO YOU WISH TO CANCEL THIS RUN ?          Y OR N ( _ )
```

4.4. INPUT, DISPLAY, SCREEN FORMAT SERVICES

4.4.1. Display and Input Using @DSP Run Function

When the @DSP run function executes, the specified report or result appears on the workstation screen.

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>          >
.DATE  83/07/05  10:47:50  TYPE=B  RID=002  83/06/29  JDOE        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.========.======.======.=====.====.======.======.======.=====.===.
 IP 741224 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 IP 741225 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 IP 741219 LS BLACKBOX2 637071        84353 INTR 741218 741219
 OR 750110 LS BLACKBOX4               94754 ARCO
 SC 750110 LS BLACKBOX5 675281        97441 FEDS 750131
 IP 741222 LS BLACKBOX5 737582        84040 AMCO 741222 741222
 SH 741203 LS BLACKBOX0 746327        54237 FEDS 741201 741202 741203 S8738
```

NOTE:

In this example, @DSP,102,B,2,,,, N is executed.

When you specify the *wait input after display* (,,,,N) parameter, you can manipulate the displayed report or result with the following manual functions:

1. Line 0 functions:

   LINE▷ FMT▷ RL▷ SHFT▷ HLD CHR▷ HLD LN▷

2. L, LL, CC, and FLD functions

3. Line update functions:

   Update line

   Add line

   Delete line

   Duplicate line

4. Run ending functions:

   Release function (∧) – Run ends and the user logo is displayed.

   Sign-off function (X) – Run ends and the idle logo is displayed.

When you input RSM after the start of entry (SOE) of line 0 or at the home position, or you press the function key F1, the run restarts execution at the next run function.

```
LINE> rsm   FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>          >
.DATE  83/07/05  10:47:50  TYPE=B  RID=002  83/06/29  JDOE        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 IP 741224 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 IP 741225 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 IP 741219 LS BLACKBOX2 637071        84353 INTR 741218 741219
 OR 750110 LS BLACKBOX4               94754 ARCO
 SC 750110 LS BLACKBOX5 675281        97441 FEDS 750131
 IP 741222 LS BLACKBOX5 737582        84040 AMCO 741222 741222
 SH 741203 LS BLACKBOX0 746327        54237 FEDS 741201 741202 741203 S8738
```

When run execution restarts, the run logo is displayed on the first line of the screen.

```
                MAPPER80 RUN FUNCTION PROCESSING <F4...CANCEL>
.DATE  83/07/05  10:47:50  TYPE=B  RID=002  83/06/29  JDOE        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 IP 741224 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 IP 741225 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 IP 741219 LS BLACKBOX2 637071        84353 INTR 741218 741219
```

When there is no *wait input after display* (,,,,Y) parameter for the @DSP run function, one screen of the report or result specified is displayed. The run then continues execution with the next run statement and the run logo reappears on the first line of the screen.

```
                MAPPER80 RUN FUNCTION PROCESSING <F4...CANCEL>
.DATE  83/07/05  10:47:50  TYPE=B  RID=002  83/06/29  JDOE        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 IP 741224 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 IP 741225 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 IP 741219 LS BLACKBOX2 637071        84353 INTR 741218 741219
 OR 750110 LS BLACKBOX4               94754 ARCO
 SC 750110 LS BLACKBOX5 675281        97441 FEDS 750131
 IP 741222 LS BLACKBOX5 737582        84040 AMCO 741222 741222
 SH 741203 LS BLACKBOX0 746327        54237 FEDS 741201 741202 741203 S8738
```

NOTE:

In this example, @DSP,102,B,2,,,,Y is executed.

4.4.2. Display and Input Using the @SFS Run Function

When the @SFS run function executes, the specified screen format services screen will be displayed at the workstation. The output data specified with the @SFS run function is displayed in the output fields and the bidirectional (input/output) fields of the screen.

If the specified screen format services screen has input fields or input/output fields, it waits for data input. Key in the data and transmit. The transmitted data enters the variables specified by the @SFS run function. The run restarts at the next run function and the run logo appears.

When there are no input fields and no input/output fields, the screen format services screen appears, then the run continues execution at the next run function. The run logo appears with the screen format services screen.

4.5. RUN END DISPLAY

4.5.1. Using the @GTO END Run Function

The run ends when the @GTO END run function executes, and the output area appears on the screen as the result. The display format is the same as that of a regular result.

Example:

```
@BRK,102,B
RUN TEST START
RUN TEST END
@GTO END
```

When these statements execute, the following information appears on the screen.

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD> ''RESULT''>
.DATE  83/07/13  14:38:15  TYPE=C  RID=     83/06/29  JDOE          <    4 LINES>
RUN TEST START
RUN TEST END
                    ..... END REPORT .....
```

If the mode number of the *displayed result is the same* as the mode number assigned when the user signed on, or the same as the mode number selected with the mode (M) manual function before execution of the run, regular manual functions can be executed for this result.

When the mode number of the *displayed result differs,* you can use only limited manual functions.

If the following statements are executed when the mode at the time of sign-on is 102, the mode of the displayed result will be 110.

Example:

```
@BRK,110,B
RUN TEST START
RUN TEST END
@GTO END
```

4.5.2. Using the @REL Run Function to End a Run

When the @REL run function executes, the run ends. The user logo appears on the screen at this time.

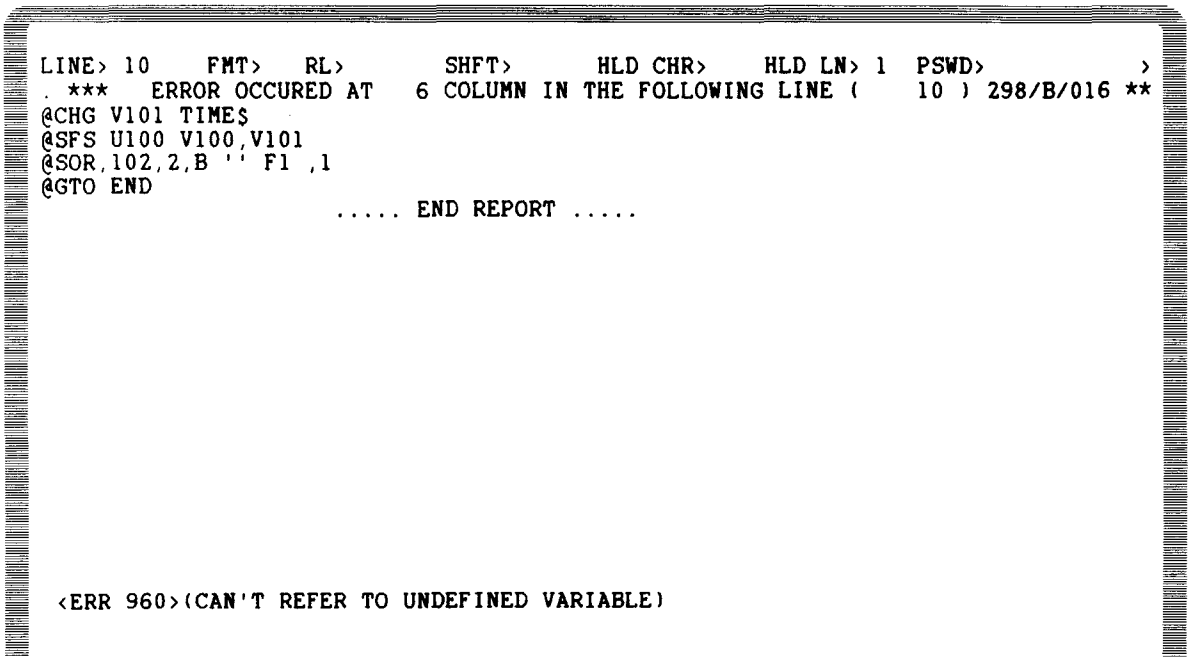
4.5.3. Using the @XIT Run Function to End a Run

When @XIT run function executes, the run ends. The idle logo appears on the screen at this time.

4.5.4. Error End of Run

When an error occurs during run execution, the run ends.

If the run ends in error, the run report appears on the screen. At the same time, an error message appears on the bottom line of the screen. Information indicating the error position (line number) and column number in the run report) also appears as a hold line on the second line of the screen.



If the mode number of the run report is the same as the user's mode number, you can immediately correct the error portion of the run report and reexecute the same run.

When the mode number of the run report differs from the user's mode number, corrections are not possible and you can use only the PR, AUX, and COP manual functions.

5. MAPPER 80 Run Function Commands

5.1. GENERAL

This section alphabetically lists the run functions. A description of each run function, its format, and examples are provided in this section. Optional parameters are enclosed by brackets.

These run functions can be entered in uppercase or lowercase letters; however, the MAPPER 80 software stores and outputs data in uppercase only.

5.1.1. @ADD (Append Report) Run Function

The @ADD run function appends a specified report (added report) to another report (the base report), creating a result.

The mode and type of the result are the same as the mode and type of the base report.

To save the result, use the @RNM, @REP, or @DUP run functions.

NOTE:

The line length of the base and added reports must be the same.

Format:

```
@ADD,mode1,type1,rid1,mode2,type2,rid2
```

where:

`mode1`
Specifies the mode of the added report.

`type1`
Specifies the type (A–I) of the added report.

`rid1`
Specifies the RID of the added report.

mode2
Specifies the mode of the base report.

type2
Specifies the type (A–I) of the base report.

rid2
Specifies the RID of the base report.

Example:

```
@ADD,2,B,2,102,B,1
```

Add 2,B,2 to 102,B,1 to create a new result.

```
@DUP,102,B,-0
```

Duplicate the result to the first available RID. The reserved word STAT1 contains the RID number of the duplicated result.

5.1.2. @ADR (Add Report) Run Function

The @ADR run function creates a new report in the mode, type, and RID number specified. When you don't specify a RID number, the new report is created with the next available report identification number. The new report's RID number is set in the reserved word STAT1.

Format:

```
@ADR,mode,type[,rid]
```

where:

mode
Specifies the mode of the new report.

type
Specifies the type (A–I) of the new report.

rid
Specifies the report number (RID) of the new report.

Example 1:

Case where the report identification number (RID) is omitted:

```
@ADR,102,B
```

New report created in mode 102 and type B.

```
@CHG V1I3 STAT1
```

Variable 1 acquires the RID number of the new report from STAT1 reserved word.

Example 2:

Case where the report identification number (RID) is specified:

```
@ADR,102,B,20
```

Creates a new report in mode 102, type B, and RID 20.

5.1.3. @AUX (Auxiliary Device) Run Function

The @AUX run function prints the specified report on any printer connected to a workstation that is linked to the same MAPPER 80 job.

Format:

```
@AUX,mode,type,rid,[station],,[sequence],[format],,[delete-header],
     [first-column],[spacing],[start-column]
```

where:

`mode`
Identifies the mode of the report to be printed.

`type`
Identifies the report type (A–I) to be printed.

`rid`
Identifies the report identification number to be printed.

`station`
Names the auxiliary equipment where the reports are to be printed. The format is Pxx.

If omitted, the report is printed on the auxiliary device connected to the workstation that started the run.

`sequence`
When a Y is specified, the line numbers are not printed.

If omitted, the line numbers are printed.

format
Specifies the format number (0–6) used when printing.

If omitted, the basic format number 0 is used.

delete-header
When Y is specified, the header lines of the report are deleted.

If omitted, the header lines are printed.

first-column
When Y is specified, the character in the first column of each line is not printed.

If omitted, printing starts with the first column.

spacing
Specifies the spacing requirements (1–9).

If omitted, the report is single-spaced.

start-column
Specifies the first column number to be printed (1–132).

If omitted, the default value is 1. When both the start-column and first-column parameters are specified, the first-column parameter is ignored.

Example:

```
@AUX,102,B,2,P01,,Y,1,,,Y,2
```

Prints mode 102, type B, and RID number 2 report on the P01 printer, deletes line number, uses format 1, deletes the first character of each report line, and double spaces.

5.1.4. @BFN (Binary Find) Run Function

The @BFN run function searches the report for the specified item by the binary division method and enters in a variable the line number containing that item. The field containing the item must be presorted in ascending order. The values of two variables are set as a direct result of this run function. They are:

variable1 Specifies the RID number of the found item.

variable2 Specifies the line number of the found item.

When the search item cannot be found, or if the specified field has not been sorted, control transfers to the specified label. An error status appears in the reserved word STAT1. The following values are set in the contents of the reserved word STAT1:

Value	Meaning
1	Specified item cannot be found.
2	Indicates that all lines are blank, when @ option is specified.
3	Indicates that data is not sorted.
4	Specified report does not have an index. See message ERR 719 for corrective action.

Format:

```
@BFN,mode,type,rid,[start-line],[label] options characters [line-type],
     parameters [variable1],variable2
```

where:

`mode`
Specifies the mode of the report to be searched.

`type`
Specifies the type (A–I) of the report to be searched.

`rid`
Specifies the RID number of the report to be searched.

`start-line`
Specifies the line number on which searching is to start.

If omitted, searching starts on the first line.

`label`
Specifies the destination label if an error occurs.

If omitted, the next run function is executed.

`options`
Specifies options. Refer to Appendix B.

`characters`
Specifies the field, partial field, or character positions to scan.

`line-type`
Specifies the line type (tab type only) to scan.

If omitted, tab type lines are assumed.

`parameters`
Specifies data (parameter) to be found.

`variable1`
Specifies the variable to hold the RID number.

`variable2`
Specifies the variable to hold the line number where the item was found.

Example:

```
@BFN,102,B,3,,99 '' F4 ,blackbox4 ,V2I3
```

Find blackbox4 in mode 102, type B, RID 3, field number 4. Enter the line number where found into variable V2.

NOTES:

1. *The branch end label, when an error occurs, is 99.*

2. *If the field to be scanned is not in ascending order, the result is unpredictable.*

The previous example is the same as entering the following data in the mask of the BF manual function.

Example:

```
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 ** ****** ** ********* ****** ****** ***** **** ****** ****** ****** ***** ***
              blackbox4
```

5.1.5. @BRK (Break) Run Function

When a run encounters an @BRK statement, it places all data in the output area into a scratch result report (–0) and clears the output area.

The mode and form type of the output area are the same as those of the run report. The mode and form type of the output area can be changed with the @BRK run function and not affect the current result. However, it does change the mode and type of the output area for the subsequent results created by the @BRK run function.

Format:

```
@BRK[,mode,type]
```

where:

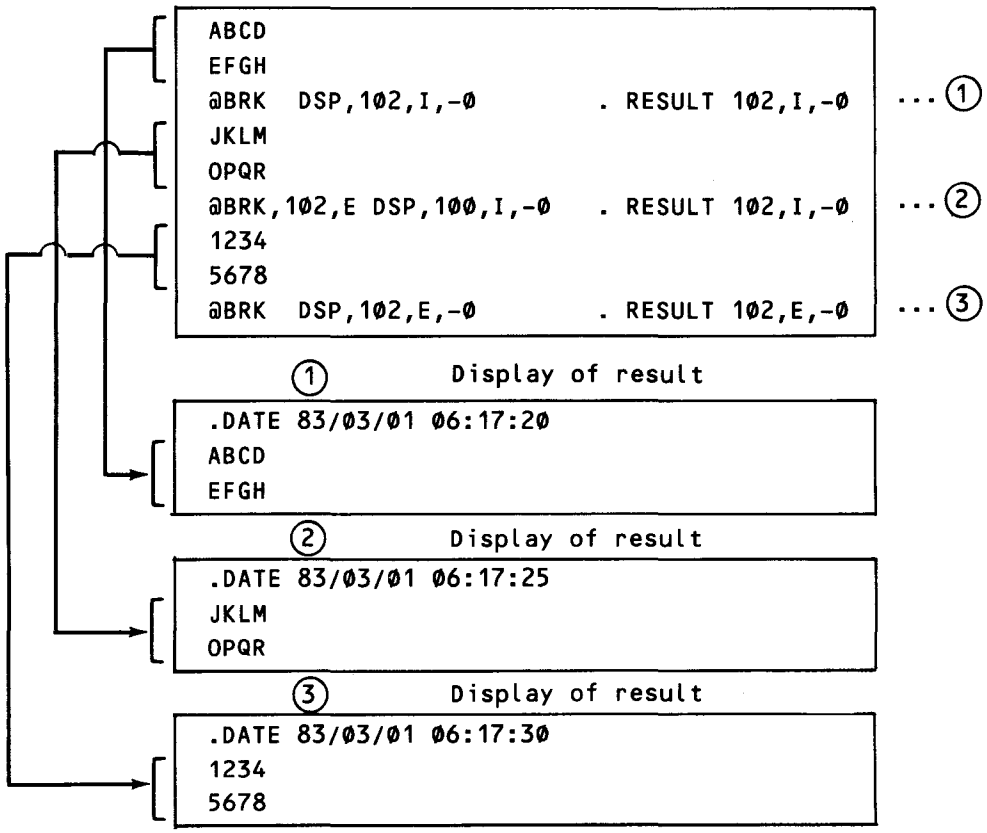
`mode`
Specifies the mode where results subsequent to the current result are placed.

`type`
Specifies the type where results subsequent to the current result are placed.

If omitted, the same mode and type as the run report, or the last @BRK that specified a mode and type, is assigned to the output area.

In the following example, the run control report is mode 102, type I. The run contains three @BRK run function statements, each of which generates a result. The first two results are assigned the mode and type of the run, 102,I. The third result is assigned the mode and type contained in the second @BRK run function, 102,E.

Example:



NOTE:

Since the MAPPER 80 software automatically sets the date in line 1, processing begins on line 2 when reading and writing the result.

5.1.6. @CHG (Change) Run Function

The @CHG run function can perform three possible assignments. These include:

1. Initializing and redefining the contents of variables with character strings, other variables, and reserved words.

2. Performing arithmetic operations and entering the result in a variable.

3. Initializing variables with run parameters passed in the reserved word INPUT$.

NOTE:

Only variable types A, F, and I can be used in arithmetic operations and as variables for entering results.

Format 1: Initializing variables

```
@CHG variable data
```

where:

`variable`
Specifies the variable in which to enter data.

`data`
Specifies the data to be entered into the variable. A character string, another variable, or reserved word can be specified.

Examples:

```
@CHG V1H3 ABC
```

Enters character string ABC in V1.

```
@CHG V2I5 1Ø2
```

Enters 102 in V2.

```
@CHG V3A5 V2
```

Enters value of variable V2 in V3.

```
@CHG V4I5 STAT1
```

Enters the value of reserved word STAT1 in V4.

Format 2: Arithmetic operation processing

```
@CHG variable expressions
```

where:

variable
Specifies a variable for entering a result of an arithmetic operation.

expressions
Describes the arithmetic expressions. The following guidelines must be followed when describing expressions:

1. Blank characters are required in front of and behind the operators +, -, *.
2. The slash symbol (/) indicates division and must be enclosed in apostrophes with a blank character before and after the apostrophes.

   Example:

   ```
   Δ '/' Δ
   ```

3. Operating sequence is from left to right.
4. The operating sequence cannot be changed by using parentheses. Parentheses are treated as an error.

Example:

```
@CHG V1I5 V2 + 1Ø
```

Enters the sum of the contents of V2 and 10 in V1.

```
@CHG V1F6.2 V2 '/' V3
```

Divide contents of V2 by contents of V3 and enter the result in V1.

```
@CHG V1A5 V4 * V5
```

Multiply contents of V4 by contents of V5 and enter the product in V1.

Format 3: Data acquired from INPUT$

```
@CHG INPUT$ variables
```

where:

INPUT$
Specifies the reserved word containing the run parameters.

`variables`
Specifies variables in which data is entered. Multiple variables can be defined with each variable being delimited with a comma.

Example:

```
@CHG INPUT$ V1A3,V2I3,V4A5
```

Run parameters are entered in V1, V2, and V4.

5.1.7. @COP (COP Device) Run Function

The @COP run function prints out the reports or results on a COP printer connected to a workstation.

Format:

```
@COP,mode,type,rid,,,[sequence],[format],,[delete-header],[first-column],
  [spacing],[start-column]
```

where:

`mode`
Specifies the mode of the report to be printed.

`type`
Specifies the type (A–I) of the report to be printed.

`rid`
Specifies the report number to be printed.

`sequence`
When Y is specified, line numbers are not printed.

If omitted, the line numbers are printed.

`format`
Specifies the format number (0–6) used when printing.

If omitted, the basic format number (0) is used.

`delete-header`
When Y is specified, the header lines of the report are deleted.

If omitted, the header lines are printed.

`first-column`
When Y is specified, the character in the first column of each line is not printed.

If omitted, the first column is printed.

`spacing`
Specifies the number of lines to space (1–9).

If omitted, the report is single-spaced.

`start-column`
Specifies the first column number to be printed (1–132).

If omitted, the default value is 1. When both the start-column and first-column parameters are specified, the first-column parameter is ignored.

Example:

```
@COP,102,D,1,,,Y,1,,,Y,2
```

Request for printout of mode 102, type D and RID 1 report on this workstation's printer. When printing, it deletes header lines, uses format number 1, deletes the first column of each line, and double spaces.

5.1.8. @DEL (Delete Lines) Run Function

The @DEL run function deletes lines found as a result of an update function such as match update (MAU) and search update (SRU).

Format:

```
@DEL
```

Example:

```
SRU,102,E '' F1 ,OVER
```

Create an update result with lines containing OVER in the first-field of the report (102,E,2).

```
@DEL
```

Delete all corresponding lines in the update result from the original report (102,E,2).

5.1.9. @DLR (Delete Report) Run Function

The @DLR run function deletes the specified report from the MAPPER 80 data base. It is not necessary to display the report before deleting it. Password restrictions are disregarded.

Format:

```
@DLR,mode,type,rid
```

where:

`mode`
Specifies the mode of the report for deletion.

`type`
Specifies the type of the report for deletion.

`rid`
Specifies the report number for deletion.

5.1.10. @DSP (Display Report) Run Function

The @DSP run function displays the specified report at the workstation in the following two ways:

1. Displays the report at the workstation and suspends the run until the manual function RSM is entered or function key (F1) is pressed; then the run restarts.

2. Display the report at the workstation and continues processing of run. The display does not change until another @DSP run function is encountered. The run logo is displayed on the first line of the screen.

The @DSP run function can be inserted into specific locations in the run for checking intermediate results when debugging.

Manual updating can also be performed, since the display appears in the form of a report or result. Manual functions that can be used are SOE update; add, delete or duplicate lines; line 0 functions; and redisplay line (L and LL).

To restart the run, enter RSM or press the function key (F1).

Format:

```
@DSP,mode,type,rid,[start-line],,[format],[interim]
```

where:

`mode`
Specifies the mode of the report to be displayed.

`type`
Specifies the type of the report to be displayed.

`rid`
Specifies the RID of the report to be displayed.

`start-line`
Specifies the first line number to be displayed.

If omitted, the display starts from line 1.

`format`
Specifies the format number used when displaying.

If omitted the basic format number (0) is used.

`interim`
When Y is specified, the run continues processing without being suspended after displaying the report.

If omitted or if N is specified, the run suspends until there is a RSM or F1 key input after displaying the report.

NOTE:

Other run statements are not allowed on the same line after the @DSP run function.

Example: When awaiting input after displaying

```
@DSP,102,D,1,,,1
```

Displays from the first line of the report in format 1. After displaying it, the run is suspended until RSM is entered on line 0 and the transmit key is pressed, or the F1 key is pressed.

Example: When continuing the run after displaying

```
@DSP,102,D,1,,,1,Y
```

Same as the previous example, but the run continues processing after displaying one screen.

5.1.11. @DUP (Duplicate Report) Run Function

The @DUP run function duplicates the specified report and places it into the same form type as the original report.

The RID number of the duplicated report is set in the reserved word STAT1.

Format:

```
@DUP,mode,type,rid
```

where:

`mode`
Specifies the mode of the report to be duplicated.

`type`
Specifies the type of the report to be duplicated.

`rid`
Specifies the report number to be duplicated.

Example:

```
@DUP,102,C,1
```

Duplicates mode 102, type C, RID 1 report.

```
@CHG V1I3 STAT1
```

Initialize V1 to contain the RID number of the new report.

5.1.12. @ESR (Exit Subroutine) Run Function

The @ESR run function is used to exit from a subroutine. When the @ESR run function command is executed, control is returned to the line following the @RSR command that called this subroutine. If a bias (+n or −n) is specified for the @ESR, the subroutine is returned to the line that is n lines after or n lines before the line that follows the @RSR run function command.

Format:

```
@ESR[,bias]
```

where:

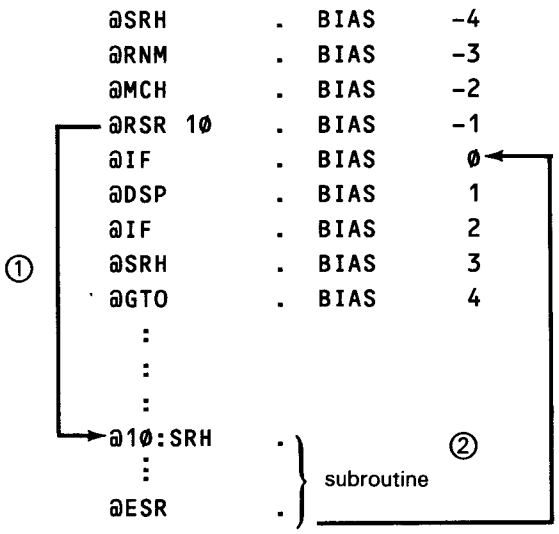
`bias`
Specifies the return line number in the format +n (or n), or −n. This bias is determined by adding to (+n) or subtracting (−n) from the line after the @ESR that called the subroutine.

If omitted, the bias is assumed to be 0, and it returns to the line after the @RSR run function command.

NOTES:

1. *If a bias is specified, the return line must be within the range of the run report.*
2. *Other run statements are not allowed on the same line after the @ESR run function.*

Example:



① Calls subroutine from label 10.

② Returns to the line, after the line it was called from.

5.1.13. @FND (Find) Run Function

The @FND run function searches for the specified item in a report and enters the first line number where the item was found into the variable. This run function sets values in the following two variables:

variable1 Specifies the RID number where the specified item was found.

variable2 Specifies the line number where the specified item was found.

When the specified item cannot be found, control is transferred to the label specified. If the label is omitted, the next instruction in the run is executed.

Format:

```
@FND,mode,type,[rid],[start-line],[label] options characters [line-type],
  parameters [variable1],variable2
```

where:

`mode`
Specifies the mode of the report to be searched.

`type`
Specifies the type of the report to be searched.

`rid`
Specifies the report number to be searched.

If RID is omitted, all RIDs in the type are processed.

`start-line`
Specifies the line-number at which to start searching the report.

If omitted, the search starts on line 1.

`label`
Specifies the destination label, if the item is not found.

If omitted, the next run function is executed.

`options`
Specifies the options. Refer to Appendix B.

`characters`
Specifies the search field, partial field, or characters.

`line-type`
Specifies the line-type (tab or *) to search.

If omitted, tab type lines are searched.

`parameters`
Specifies the find parameters.

`variable1`
Specifies the variable where the RID number of the found item is entered.

`variable2`
Specifies the variable where the line-number of the found item is entered.

Example:

```
@FND,102,B,2,,99 '' F4 ,blackbox4 ,V2I5.
```

Search report 102,B,2 field 4 for blackbox4 and enter the line-number where found into variable2.

If not found, go to label 99.

5.1.14. @GTO (Branch) Run Function

The @GTO run function is used in two ways:

1. Branches to the specified line (run function) during the run.

2. Terminates the run, closes the output area, and displays the output area as a result.

Format 1: Branching within a run

```
@GTO label
```

where:

label
Specifies the end of branch. The following four formats are possible:

label	Is a label number from 1 through 199.
variable	Is a variable containing a label number from 1 through 199.
LIN+n	Is the number of lines following the present line. The n cannot be expressed as a variable; n can be a number between 1 and the range of the report.
LIN–n	Is the number of lines preceding the present line. The n cannot be expressed as a variable; n can be a number between 1 and the range of the report.

Format 2: End of run

```
@GTO END
```

where:

END
Specifies the end of the run.

Examples:

```
@GTO 1Ø
```

Branches to the line labeled 10.

```
@CHG V1I3 12Ø
@GTO V1
```

Branches to the line labeled 120.

```
@GTO LIN+5
```

Branches 5 lines past *this line*.

```
@GTO LIN-10
```

Branches 10 lines in front of this line.

```
@GTO END
```

Ends the run.

5.1.15. @IF (Conditional) Run Function

The @IF run function compares values and makes logical decisions based on the results. The types of comparisons are:

- Simple comparison of equal, greater than, or less than
- Equal value comparison with multiple values (OR conditional)
- Range comparison

Format 1: Simple comparison

```
@IF variable operator value action1 [;action2]
```

where:

`variable`
Specifies the variable for comparison. The maximum length is 64 characters.

`operator`
The following operators are valid for simple comparison:

Operator	Meaning
= or EQ	Equal
> or GT	Greater than
< or LT	Less than
NOT = or NOT EQ	Not equal
NOT > or NOT GT	Not greater than
NOT < or NOT LT	Not less than

`value`
Describes data for comparison. Values can be constants, character strings, or variables.

`action1`
Specifies the run function to be executed when the comparison is equal (true). If action1 does not contain an @GTO run function then action2 is executed.

action2
Specifies the run function to be executed when the comparison is not equal (false).

If action2 is omitted, not equal conditions fall through and execute the next run function command.

NOTE:

Action1 and action2 can be compound instructions. The space character separating action1 from action2 is not required in any of the @IF run function formats. Format 2 is an example where the space character is omitted.

```
@IF V1 = V2 GTO 1;GTO 2
```

Branches to label 1 if V1 is equal to V2; otherwise, branch to label 2.

Format 2: Equal value comparison with multiple values (OR condition)

```
@IF variable operator value1,value2,...,valuen action1 [;action2]
```

where:

variable
Specifies the value to be compared.

operator
Always specifies the = (equal) operator.

value1,value2,...,valuen
Specifies the values for comparison with variable.

action1
Specifies the run function to be executed, if the result is true.

action2
Specifies the run function to be executed, if the result is false.

If action2 is omitted and the result is false, the next run function is executed.

NOTE:

The result is true, if the variable is equal to any of the values; otherwise, the result is false.

Example:

```
@IF V1 = V2,V3,V4 GTO 1
```

Branches to label1 if the contents of V1 is equal to the contents of V2, V3, or V4; otherwise, execute the command on the next line.

Format 3: Range comparison

```
@IF variable operator1 value1 { < & > } operator2 value2 action1 [;action2]
                              { <AND> }
```

NOTES:

1. *The following operators are valid for range comparisons:*

   > or GT *Greater than*

   < or LT *Less than*

   & or AND *Conjunction of statements*

2. *When operator1 is GT, then operator2 must be LT.*

3. *When operator1 is LT, then operator2 must be GT.*

where:

`variable`
Specifies the value to be compared.

`operator1`
Specifies GT or LT.

`value1`
Specifies the first value to be compared against the variable.

`AND (&)`
Specifies the connective to the second part of the comparison.

`operator2`
Specifies GT or LT.

`value2`
Specifies the second value to be compared against the variable.

`action1`
Specifies the run function to execute, if the result is true.

`action2`
Specifies the run function to execute, if the result is false.

If action2 is omitted, and the result is false, the next run function is executed.

NOTE:

The result is true if the variable meets the conditions stated by operator1 & value1 AND operator2 & value2; otherwise, the result is false.

Example:

```
@IF V1 > 100 & < 200 GTO 1
@IF V1 < 50 CHG V1 V1 + 1
```

Branch to label 1 if V1 is greater than 100 and less than 200; otherwise, execute the command on the next line that adds 1 to V1, if V1 is less than 50.

5.1.16. @IND (Index) Run Function

The @IND run function reads a specified number of lines from each report in the selected mode and type and then creates a result (index).

Values are set in two reserved words:

STAT1 Specifies the number of reports in the type.

STAT2 Specifies the total number of lines in all reports in the type.

Format:

```
@IND,mode,type,lines[,label][,header]
```

where:

`mode`
Specifies the mode of the report to index.

`type`
Specifies the type of the report to index.

`lines`
Specifies the number of lines (1 through 999) to be read from each report.

`label`
Specifies the destination label, if no RID exists in the designated mode and type.

If omitted, the next run function is executed.

`header`
When H is specified, the header lines after the first RID are not to be included in the index.

Example:

`@IND,102,B,5,20,H`
Extracts five lines from each report in mode 102, type B. Do not include report headers from second and subsequent RIDs. If no RIDs are found, branch to label 20.

5.1.17. @INS (Insert) Run Function

The @INS run function inserts data from one variable to another or inserts constant data into a variable.

Format:

```
@INS data variable
```

where:

`data`
Describes the data to be inserted in the variable. Constants, character strings, and variables can be used.

`variable`
Specifies the variable, the position in the variable and the number of characters in the format: Vn(m–l), where:

Vn Is the variable number.

m Is the position in the variable.

l Is the number of characters. The m and l can be variables.

Example:

```
@CHG V1H2 QQ CHG V2H3 AAA
```
Initialize V1 to QQ and V2 to AAA.

```
@INS V1  V2(2-2)
```
Insert the contents of V1 into V2 at column 2 for two characters. V2 now equals AQQ.

```
@CHG V4H6 123456 CHG V7I1 3
```
Initialize V4 to 123456 and V7 to 3.

```
@INS ABCD V4(V7-4)
```
Insert ABCD into V4 starting at column V7 for four characters. V4 now equals 12ABCD.

```
@CHG V5H6 SSSSSS CHG V3I1 2
```
Initialize V5 to SSSSSS and V3 to 2.

```
@INS V4(V3-V7) V5(2-V7)
```
Insert the contents of V4 (starting at column V3 for V7 characters) into V5 (starting at column 2 for V7 characters). V5 now equals S2ABSS.

5.1.18. @LCH (Locate and Change) Run Function

The @LCH run function scans a report in the specified column character positions looking for the target string. On each occurrence of the target string, it changes the target string in the report to the replacement string. The @LCH run function produces a result.

Format:

```
@LCH,mode,type,rid,[start-line],[label] option characters target-string
/replace-string [,variable2,variable3]
```

where:

`mode`
Specifies the mode of the report to scan.

`type`
Specifies the type of the report to scan.

`rid`
Specifies the report number to scan.

`start-line`
Specifies the line-number at which to start.

`label`
Specifies the destination label when the target string cannot be found.

If omitted, the next run function is executed.

`options`
Specifies the options. Refer to Appendix B.

`characters`
Specifies the columns, fields, or partial fields to be scanned.

`target-string`
Specifies the target string (maximum of 32 characters).

`replace-string`
Specifies the replacement string (maximum of 32 characters).

`variable2`
Specifies the variable for entering the number of lines that contained the target string.

`variable3`
Specifies the variable for entering the RID number where the target string was located.

In the @LCH run function, the relationship between the options and the target-string and replace-string is as shown in Table 5–1.

Table 5–1. Target-String/Replace-String Relationship

Options	Target-String		Replace-String	
	First character	Second character	First character	Second character
No	Line type to be scanned (tab or *)	First character of target-string	Ignored, but must contain fill character	First character of replace-string
A	Ignored, but must contain fill character	First character of target-string	Ignored, but must contain fill character	First character of replace-string
F	target-string		replace-string	
M	Line type to be scanned	First character of target-string	New line type when replacment occurs	First character of replace-string

Examples:

```
@LCH,mode,type,rid '' 1-128 ]ABC/Ⓣ DEF ,V2,V3
```

Only tab lines are scanned.

```
@LCH,mode,type,rid A 1-128 Ⓣ ABC/Ⓣ DEF ,V2,V3
```

All lines are scanned.

```
@LCH,mode,type,rid F 1-128 ABC/DEF ,V2,V3
```

All lines are scanned.

```
@LCH,mode,type,rid M 1-128 ]ABC/*DEF ,V2,V3
```

Only tab lines scanned. Line type is changed to * when replacement has occurred.

NOTES:

1. *The Ⓣ is a fill character.*
2. *If your target-string includes spaces, use the Ⓣ fill character for assigning the transparent character to another character value.*

5.1.19. @LLN (Last Line Number) Run Function

The @LLN run function sets a variable equal to the last line number of the specified report.

Format:

```
@LLN,mode,type,rid[,label] variable
```

where:

mode
Specifies the mode of the report to be checked.

type
Specifies the type of the report to be checked.

rid
Specifies the report number to be checked.

label
Specifies the destination label when the specified report does not exist.

If omitted, the next run function is executed.

variable
Specifies the variable in which to enter the last line number of the report.

Example:

```
@LLN,102,B,1,99 V1I3
```

The last line number of the report is entered in variable V1.

5.1.20. @LNX (Duplicate Line) Run Function

The @LNX run function duplicates lines in the report. The maximum number of lines that can be duplicated is 23. The duplicate quantity parameter allows lines to be duplicated from 1 to 99 times. For example, 1 line can be duplicated 99 times, 23 lines can be duplicated 4 times, or any other combination within the guidelines (lines to be duplicated can equal a maximum of 23, while duplicated lines can equal 99.)

Before executing this instruction, you must lock this report with the @LOK run function command.

Format:

```
@LNX,mode,type,rid,start-line,duplicate-quantity[,size]
```

where:

mode
Specifies the mode of the report requiring duplication.

`type`
Specifies the type of the report requiring duplication.

`rid`
Specifies the report number requiring duplication.

`start-line`
Specifies the first line number to be duplicated.

`duplicate-quantity`
Specifies the number (n) of times that the same line or lines is to be duplicated.

`size`
Specifies the number of lines (p) up to 23 to be duplicated. The default value is 1 and (n x p) must be less than or equal to 99.

Examples:

```
@LNX,102,B,10,10,5
```

Duplicates the tenth line five times.

```
@LNX,102,B,30,20,4,5
```

Duplicates five lines starting with line 20 four times. A total of 20 lines are duplicated in the report.

5.1.21. @LN+ (Add Line) Run Function

The @LN+ run function adds lines to the report. The line defined in RID 0 of the specified report type is used as the line to be added. A maximum of 99 lines can be added with one @LN+ run function.

Lock this report prior to executing the @LN+ run function command.

Format:

```
@LN+,mode,type,rid,start-line,line-quantity [,line-number]
```

where:

`mode`
Specifies the mode of the report to which the line is added.

`type`
Specifies the type of the report to which the line is added.

`rid`
Specifies the report number to which the line is added.

`start-line`
Specifies the line number after which lines are added.

`line-quantity`
Specifies the number of lines to add. A maximum of 99 lines can be added.

`line-number`
Specifies the predefined line number (0 through 9) to be added. The default is 0 (normally a blank tab line).

Example:

```
@LN+,102,C,5,40,10,2
```

Add 10 lines of predefined line number 2 from RID 0 after line 40.

5.1.22. @LN– (Delete Line) Run Function

The @LN– run function deletes from 1 to 999 lines from a report. The @LN– must be preceded by an @LOK run function command.

Format:

```
@LN-,mode,type,rid,start-line,line-quantity
```

where:

`mode`
Specifies the mode of the report having lines to be deleted.

`type`
Specifies the type of the report having lines to be deleted.

`rid`
Specifies the report number having lines to be deleted.

`start-line`
Specifies the first line number to be deleted.

`line-quantity`
Specifies number of lines to be deleted.

Example:

```
@LN-,102,C,5,41,10
```

Deletes 10 lines starting at line 41.

5.1.23. @LOK (Update Lock) Run Function

The @LOK run function temporarily prevents other users from updating the report currently being updated. Use an @LOK command before an @LNX, @LN+, @LN–, or a @WRL run function, or else the run ends in error.

If another user already has update control of the report, the run stalls until that user releases control.

The lock can be released by the following actions:

1. Execution of the @ULK run function
2. Execution of the @LOK run function on other reports
3. Execution of the following run functions:

```
@ADD, @ADR, @AUX, @BFN, @BRK, @COP, @DEL, @DLR, @DSP, @DUP, @FND, @IND,
@LCH, @LLN, @MAU, @MCH, @PRT, @RDC, @RDL, @REP, @RLN, @RNM, @RUN,
@SOR, @SRH, @SRU, @SUB, @TOT, @UPD
```

4. End of run functions: @REL, @XIT, @GTO END or when a run ends in error

Format:

```
@LOK,mode,type,rid
```

where:

`mode`
Specifies the mode of the report to be locked.

`type`
Specifies the type of the report to be locked.

`rid`
Specifies the RID of the report to be locked.

Example:

```
@LOK,102,D,1
```

The mode 102, type D, RID1 report is locked.

5.1.24. @MCH (Match) Run Function

The @MCH run function compares fields in two different reports. When a match is detected, it transfers the contents of a specified field in the issuing report to the specified field of the result. The result produced by this report is a copy of the receiving report with the transfer fields blanked out or, updated with the data transferred from the issuing report. The contents of the result will vary depending on the options you specify. When the P option is specified, the following values are assigned in the two reserved words:

STAT1 Specifies the number of matched lines or number of unmatched lines when the N option is specified.

STAT2 Specifies the number of lines of the receiving report on which a match was attempted.

Format:

```
@MCH,mode1,type1,rid1,mode2,type2,rid2,[label] options characters1
[line-type1],parameters1 characters2 [line-type2],parameters2
```

where:

`mode1`
Specifies the mode of the issuing report.

`type1`
Specifies the type of the issuing report.

`rid1`
Specifies the report number of the issuing report.

`mode2`
Specifies the mode of the receiving report.

`type2`
Specifies the type of the receiving report.

`rid2`
Specifies the report number of the receiving report.

`label`
Specifies the destination label when the comparison fields have not been sorted and the P option was specified.

`options`
Specifies the options. Refer to Appendix B.

`characters1`
Specifies the comparison fields and the transfer fields of the issuing report. A maximum of 5 comparison fields with a maximum of 36 characters can be specified. A maximum of 13 transfer fields can be specified.

`line-type1`
Specifies the line type (tab) to be processed by the issuing report.

`parameters1`
Specifies the comparison fields (1 through 5) and the data transfer fields (A–M) of the issuing report.

`characters2`
Specifies the comparison fields and the data transfer fields in the receiving report.

`line-type2`
Specifies the line type (tab) of the receiving report.

`parameters2`
Specifies the comparison fields (1 through 5) and the data transfer fields (A–M) in the receiving report that correspond to parameters1 in the issuing report.

Example:

```
@MCH,102,C,1,102,B,2 '' F1,F3 ,1,A F4,F6 ,1,A
```

There are no labels or options specified in the example.

- Issuing Report

  `F1`
  Is the comparison field in the issuing report.

  `F3`
  Is the third field in the issuing report and the field from which data is transferred to the result.

  `1`
  Is the first match parameter in the issuing report. It relates to F1 and means that F1 is the first field in the issuing report to be compared to the corresponding first field in the receiving report (F4). A maximum of five fields (1 through 5) can be compared.

  `A`
  Is the first data transfer field in the issuing report. It relates to F3 indicating that, when a match occurs, the contents of F3 transfers to the corresponding A (F6) field in the result. A maximum of 13 fields (A–M) can be transferred.

- Receiving Report

  `F4`
  Is the fourth field in the receiving report. It is compared against F1, the first field in the issuing report.

F6

Is the sixth field in the receiving report. This indicates the transfer-to field that receives the transferred data from F3 of the issuing report.

1

Is the first match parameter in the receiving report. It relates to F4 and means that F4 is the first field in the receiving report to be compared to the corresponding first field in the issuing report (F1).

A

Is the first data receive field in the receiving report. It relates to F6 and indicates that, when a match is found, the contents of the corresponding A field in the issuing report (F3) is transferred to the first data transfer field (F6) in the result. If no match is found, this field is made blank.

The following is a comparative example using the MA manual function. It compares the Product Type fields of two reports. If they match, the data in the Product Cost field is transferred from the issuing report to the result.

Example of Issuing Report:

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>           >
.DATE  83/06/29  13:07:49  TYPE=C  RID=001  83/06/29  JDOE        <   24 LINES>
. <<<  CORPORATE FACTORS BASE  >>>
* PRODUCT . SUB .PRODUC. WHOLE . RETAIL . SALES .SPACE.  DEMO  .              .
*  TYPE   . KEY . COST . SALE$ .  $$$$  .COMMISS. REQ .QUANTITY. DEMO RESULTS  .
*=========.=====.======.=======.=======.=======.=====.========.===============.
 BLACKBOX1     A  13500   16875    23625 2362.50   100        1
 BLACKBOX2     A  13600   17000    23800 2380.00   110        2
 BLACKBOX3     A  13700   17125    23975 2397.50   120        4
```

Example of Receiving Report:

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>           >
.DATE  83/07/05  10:47:50  TYPE=B  RID=002  83/06/29  JDOE        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 IP 741224 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 IP 741225 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 IP 741219 LS BLACKBOX2 637071        84353 INTR 741218 741219
```

Example of Match Mask:

```
* PRODUCT . SUB .PRODUC. WHOLE . RETAIL . SALES .SPACE.  DEMO  .                .
*  TYPE   . KEY . COST . SALE$ .  $$$$  .COMMISS. REQ .QUANTITY. DEMO RESULTS  .
*=========.=====.======.=======.========.=======.=====.========.===============.
 ********* ***** ****** ******* ******** ******* ***** ******** ***************
 1             a
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 ** ****** ** ********* ****** ****** ***** **** ****** ****** ****** ***** ***
             1               a
```

5.1.25. @MAU (Match Update) Run Function

The @MAU run function performs essentially the same operation as the @MCH run function. However, using the update results generated by the MAU run function, you can replace or delete lines in the original receiving report.

Execution of the @MAU run function causes the receiving report to be locked until execution of the @UPD or @DEL run functions.

Format:

```
@MAU,mode1,type1,rid1,mode2,type2,rid2,[label] options characters1
[line-type1],parameters1 characters2 [line-type2],parameters2
```

NOTE:

Refer to the @MCH run function (5.1.24) for parameter definitions of @MAU.

Example:

```
@MAU,102,C,1,102,B,2 M F1,F3 ,1,A F4,F6 ,1,A
@UPD
```

Execution of this run statement produces an update result. This update result is exactly the same as the result produced by the @MCH example (5.1.2.4). The only difference is that the contents of this update result can be used to replace or delete corresponding lines in the original receiving report.

The @UPD run function replaces lines in the original receiving report (102, B, 2) with corresponding lines in the update result.

5.1.26. @PRT (Print) Run Function

The @PRT run function prints reports on the system printer.

Format:

```
@PRT,mode,type,rid,,[format],,,,[spacing]
```

where:

`mode`
Specifies the mode of the report to be printed.

`type`
Specifies the type of the report to be printed.

`rid`
Specifies the RID number of the report to be printed.

`format`
Specifies the format number (0 through 6) used when printing.

If omitted, it defaults to 0 (basic format number).

`spacing`
Specifies the number of line advances when printing (1 through 9). The default is 1.

Example:

```
@PRT,102,D,1,,1,,,,2
```

Print the report in format 1, double space.

5.1.27. @RDC (Read Continuous) Run Function

The @RDC run function reads a specified number of data lines continuously from the report. The data lines read must be sent to the output area with the output line located after this instruction; otherwise, the run ends in error.

Format:

```
@RDC,mode,type,rid,start-line,line-quantity,[line-type],[label]
characters variables
```

where:

`mode`
Specifies the mode of the report to be read.

`type`
Specifies the type of the report to be read.

`rid`
Specifies the report number of the report to be read.

`start-line`
Specifies the starting line number.

`line-quantity`
Specifies the number of lines to be read.

`line-type`
Specifies the line type.

If omitted, all line types are read in.

`label`
Specifies the destination label if the starting line number does not exist or if the last line number is reached during the read.

`characters`
Specifies fields, partial fields, or character columns to be read in the line.

`variables`
Specifies the variables where the data goes.

NOTES:

1. *The @RDC run function must not be followed on the same line by another run function.*

2. *The line following the @RDC run function must be an output line.*

Example:

```
@RDC,102,C,1,6,10,,40 F1,F3 V1A9,V2I6
```

Reads 10 lines starting from the sixth line of mode 102, type C, RID 1 report and writes the first and third fields into variables (V1 and V2. Product Type = V1 and Product Cost = V2.

```
PRODUCT TYPE=V1  PRODUCT COST=V2
```

Write the data to the output area.

5.1.28. @RDL (Read Line) Run Function

The @RDL run function reads in one line of data from a specified report. After executing this function, the next line number to be read is entered in the reserved word: LINE$. When the END REPORT line is read, a zero is entered in LINE$.

Format:

```
@RDL,mode,type,rid,line[,label] characters variables
```

where:

`mode`
Specifies the mode of the report.

`type`
Specifies the type of the report.

`rid`
Specifies the report number.

`line`
Specifies line number to be read.

`label`
Specifies the destination label when the line number to be read does not exist.

`characters`
Specifies the fields, partial fields, or character columns to be read in the line.

`variables`
Specifies the variables where the read in data goes.

Example:

```
@RDL,10Ø2,B,2,6,99 F1,F2,F4 V1A2,V2A6,V3A9
```

5.1.29. @REL (Release) Run Function

The @REL run function releases the workstation, terminates the run, and displays the user logo.

Format:

```
@REL
```

5.1.30. @REP (Replace) Run Function

The @REP run function replaces a receiving report with an issuing report.

On execution, the contents of both reports become identical. Result names (–0 to –4) can be specified for the issuing report but not for the receiving report.

NOTE:

The line length of both reports must be the same.

Format:

```
@REP,mode1,type1,rid1,mode2,type2,rid2
```

where:

`mode1`
Specifies the mode of the issuing report.

`type1`
Specifies the type of the issuing report.

`rid1`
Specifies the report number of the issuing report.

`mode2`
Specifies the mode of the receiving report.

`type2`
Specifies the type of the receiving report.

`rid2`
Specifies the report number of the receiving report.

Example:

```
@REP,20,E,5,40,E,10
```

Replace mode 40, type E, RID 10 with mode 20, type E, RID 5 report.

5.1.31. @RLN (Read Next Line) Run Function

The @RLN run function reads one line of data from a report. The report was specified by the @RDL run function that was executed immediately prior to the @RLN run function. The line number read is the line number specified in this run function or, if omitted, the line number contained in the reserved word LINE$. After the last line is read, LINE$ contains 0.

If certain run functions are executed between the @RDL and @RLN run functions or between an @RLN and the next @RLN run function, an error results. These run functions include:

```
@ADD, @ADR, @AUX, @BFN, @BRK, @COP, @DEL, @DLR, @DSP, @FND, @IND, @LCH, @LLN,
@LNX, @LN+, @LN-, @LOK, @MAU, @PRT, @RDC, @RDL, @REP, @RNM, @RUN, @SOR, @SRH,
@SRU, @SUB, @TOT, @ULK, @UPD, @WRL
```

Format:

```
@RLN,[line][,label] characters variables
```

where:

`line`
Specifies line number to be read.

If omitted, the line number contained in the reserved LINE$ is read.

`label`
Specifies the destination label when the specified line number does not exist.

`characters`
Specifies the fields, partial fields, or character columns to be read in the line.

`variables`
Specifies the variables that contain the data read in.

Example:

`@RDL,20,F,4,100,99 F1,F2,F3-24 V1A4,V2I4,V3S24`	Read line 100 of the mode 20, type F, RID 4 report. Enter the data in F1 into V1, F2 into V2, and the first 24 characters of F3 into V3.
`V1 V2 V3`	Write V1, V2, and V3 to the output area.
`@GTO 11`	Branch to the line labeled 11.
`@10:RLN,,99 F1,F2,F3-24 V1,V2,V3`	Read in the line contained in the reserve word LINE$ and enter the data in F1 into V1, F2 into V2, and the first 24 characters of F3 into V3.
`V1 V2 V3`	Write V1, V2, and V3 to the output area.
`@11:CHG V9I3 LINE$`	Initialize V9 with the contents of the reserved word, LINE$.
`@IF V9 < 200 GTO 10`	If the next line number is less than 200, loop back to the line labeled 10. If the next line number is equal to or greater than 200, execute the next run function.
`@99:BRK DSP,20,F,-0`	Display the output area on the screen.
`@XIT`	Terminate the run.

5.1.32. @RNM (Rename) Run Function

The @RNM run function establishes a new reference number for a report or result. This number can be: –1, –2, –3, or –4, in any order. This run function is especially useful for temporarily saving multiple results.

The same number can be used as often as you like; however, each time the number is used, its previous contents are erased. The temporarily saved data is completely erased at the end of the run. Therefore, permanent updating must be completed before the end of the run.

Renamed reports can be referenced by their renamed numbers (–1 to –4).

Format 1: Renaming the result –0

```
@RNM -n
```

where:

`-n`
Specifies the new result number (–1 to –4) of result (–0).

Example:

```
@RNM -2
@SOR,102,B,-2
```

Renames the result –0 to result –2 and then sorts it.

Format 2: Renaming the report

```
@RNM,mode,type,rid -n
```

where:

`mode`
Specifies the report to be renamed.

`type`
Specifies the type of the report to be renamed.

`rid`
Specifies the report number of the report to be renamed.

`-n`
Specifies the result number (–1 to –4) to be assigned to the report.

Example:

```
@RNM,102,C,1 -3
```

Renames the report 102,C,1 to result (–3).

5.1.33. @RSR (Run Subroutine) Run Function

The @RSR run function calls and executes subroutines within the same run report. A subroutine is a series of run functions that starts with a label and ends with the @ESR function. Return from the called subroutine is accomplished by the @ESR.

Subroutines can also call other subroutines. This is called subroutine nesting, and a maximum of 10 levels can be nested. Other run function commands must not appear on the same line after the @RSR.

Format:

```
@RSR label
```

where:

label
: Specifies the entrance label (1 through 199) of the subroutine.

Example:

```
@RSR 1                                  Execute subroutine at label 1.
@ . control statement a  .
@ . control statement b  .
@ . control statement c  .
@GTO END .                              Subroutine statements
@1: subroutine control statement a
@ . subroutine control statement b
@ . subroutine control statement c
@ESR .                                  End the subroutine and return to calling
                                        RSR + 1 line (statement a).
```

5.1.34. @RUN (Start Run) Run Function

The @RUN run function ends the run being executed (RUN1), and starts execution of a successor run (RUN2). A run cannot name itself as the successor run.

RUN1 can transfer to RUN2 a maximum of 40 different variables and/or constants (with a maximum of 320 characters). These data are entered in the reserved word INPUT$. RUN2 can access the transferred data in the reserved word INPUT$ by using the @CHG run function.

The result (–0) created in run number 1 can be used in run number 2.

Other run functions must not appear on the same line after the @RUN function.

Format:

```
@RUN run-name[,data]
```

where:

run-name
Specifies the run name of run number 2.

data
Specifies the variables and constants to be passed to run number 2. Each data segment is delimited by a comma.

Example:

```
    (RUN1)
@CHG V1H4 1234 CHG V2H3 STNUM$        Initialize variables to be passed to RUN2.
@RUN RUN2,V1,V2,TEST                  Start the RUN2.
    (RUN2)
@CHG INPUT$ V1H4,V2H3,V3H4            Initialize variables with data passed from RUN1.
@RNM -1                               Rename result (-0) from RUN1 to -1.
@BRK
```

5.1.35. @SFS (Display Screen Format) Run Function

The @SFS run function displays, at the workstation, the formatted screen that the user created with the system utility routine (screen format services). Refer to the OS/3 screen format services, concepts and facilities, UP-8802 (current version).

Fields in the formatted screens can be defined as output, input, or input/output. Output-only fields display data. Input-only fields accept input data. Input/output fields display data and accept input data.

If the screen displayed by the @SFS run function contains input/output or input-only fields, the run is temporarily suspended. This suspension continues until the data is keyed into these fields and transmitted. The input data is written into the variables specified in the @SFS input-data parameter.

If the screen contains only output fields, the next run function is executed immediately after displaying the screen.

Format:

```
@SFS format-name [output-data][/input-data]
```

where:

format-name
Specifies the SFS format name to be displayed. This must be four alphanumeric characters with the first character being the letter U.

This SFS format name must be registered in the screen format file.

output-data
Defines the output variables and constants to be inserted in the output and input/output fields.

When there is multiple data, each data segment is delimited by a comma.

The total number of characters of the data defined here must be equal to the total number of characters of the output and/or the input/output fields of the @SFS format to be displayed. A six position field must contain six characters. If this does not match, the run ends in error.

Omit this field when the screen contains no output or input/output fields.

input-data
Specifies the variables receiving the input data. When multiple variables are specified, each variable must be delimited with a comma. Variables do not have to be specified when there are no input or input/output fields in the format.

Fields in your defined screens can include any of the basic character types (A, 9, X). However, the editing characters (sign, comma, and period in a numeric field) are not supported.

Examples:

The following is a screen display of the SFS format name UAC1 and this screen is created using the system utility routine, Screen Format Services.

```
            <<< PLEASE ENTER THE REQUESTED DATA >>>
DATE: uuuuuu

                SALES PERSON      - (____)
                PRODUCT TYPE      - (____)
                ORDER QUANTITY    - (____)
```

where:

uuuuuu
Specifies output field.

____
Specifies input field.

When:

```
@CHG V1I6 DATE1$                    Initialize V1 with date.
@SFS UAC1 V1/V2A5,V3A4,V4I4         Display SFS screen UAC1.
```

are executed, the following format is displayed:

```
            <<< PLEASE ENTER THE REQUESTED DATA >>>
DATE: 830714

                SALES PERSON      - (____)
                PRODUCT TYPE      - (____)
                ORDER QUANTITY    - (____)
```

Data entered into the SALES PERSON, PRODUCT TYPE, and ORDER QUANTITY fields are set into the variables V2, V3, and V4, respectively.

Examples:

When there is no output data:

```
@SFS UA02 /V1I3,V2A3
```

When there is no input data:

```
@SFS UA03 ABC
```

When there is no input or output data:

```
@SFS UA04
```

When there is input/output data:

```
@CHG V1A6 DATE1$          Initialize V1 with the date.
@CHG V2A3 RSM             Initialize V2 with RSM. (This is an input/output field.)
@SFS UA04 V1,V2/V2        Display SFS screen UA04 and suspend the run. You can change the
                          data in the input/output field from RSM to HLT to terminate the run
                          or leave RSM in the input/output field. In either case, press the
                          transmit key to continue the run.
@IF V2= 'HLT' GTO 99      If the input/output field is HLT, go to line labeled 99;
                          otherwise, execute the next run function.
  .
  .
  .
@99:XIT                   Terminate the run.
```

5.1.36. @SOR (Sort) Run Function

The @SOR run function sorts lines in a report in ascending or descending order and produces a result. Sort keys can be specified for up to five fields. Each key can be designated as ascending (numeric or character) sequence or descending (numeric or character) sequence.

Format:

```
@SOR,mode,type,rid options characters [line-type],parameters
```

where:

`mode`
Specifies the mode of the report to be sorted.

`type`
Specifies the type of the report to be sorted.

`rid`
Specifies the report number of the report to be sorted.

`options`
No options are available. Use apostrophes as shown in the example.

`characters`
Specifies the fields to be sorted with a maximum of 5 fields and a total of 36 or fewer characters.

`line-type`
Specifies the line type (tab or *) to be processed.

`parameters`
Specifies the parameters of the sort fields (1 through 5) and A (ascending), D (descending), and N (numeric) order.

NOTES:

1. *The sort level (1 – 5) must be specified.*
2. *When the A or D parameter is not specified, the default is A.*
3. *The N parameter is used in combination with the sort level and the A or D parameters, for example, 2DN, or 2AN, or 2N.*
4. *When the N parameter is specified, the data is sorted in numeric sequence.*
5. *When the N parameter is not specified, the data is sorted according to the Series 80 EBCDIC code table (character sequence).*

Example:

```
@SOR,102,B,2 '' F2,F4,F7 ,1,3D,2N
```

Sort the second field of mode 102, type B, RID 2 report in ascending order. The sort key is at the first level and at the seventh field with the sort key at the second level by numerical value in ascending order. Then sort the fourth field in descending order with the sort key at the third level.

5.1.37. @SRH (Search) Run Function

The @SRH run function searches vertically through a report and creates a result with the lines found.

Format:

```
@SRH,mode,type,[rid],[start-line],[line-quantity],[label] options
 characters [line-type],parameters [variable1],[variable2]
```

where:

`mode`
Specifies the mode of the report to be searched.

`type`
Specifies the type of the report to be searched.

`rid`
Specifies the report number to be searched. If RID is omitted then all RIDs are searched.

`start-line`
Specifies the starting line number of the search.

`line-quantity`
Specifies the number of lines to be searched.

`label`
Specifies the destination label when specified data cannot be found.

`options`
Specifies options (Appendix B).

`characters`
Specifies the fields, partial fields, or character columns to search.

`line-type`
Specifies the line type (tab or *) to search.

parameters
Specifies the search parameters.

variable1
Specifies the variable where the number of lines found is entered.

variable2
Specifies the variable where the number of lines searched is entered.

Examples:

When the parameter is an AND condition:

```
@SRH,102,B,2 '' F1,F4 ,SC,blackbox5 V1I3,V2I4
```

When the parameter is an OR condition:

```
@SRH,102,B,2 '' F8 ,AMCO/,FEDS V1I3,V2I4
```

When the parameter is a range search:

```
@SRH,102,B,2 '' F5,F4-8 ,100000,blackbox/R,700000 V1I3,V2I4
```

5.1.38. @SRU (Search Update) Run Function

The @SRU run function searches vertically through a report and creates an update result with the lines found. The lines of the original report corresponding to the lines of the update result can be deleted with the @DEL run function or updated with the @UPD run function.

Format:

```
@SRU,mode,type,rid,[start-line],[line-quantity],[label] options
 characters [line-type],parameters [variable1],[variable2]
```

NOTE:

Refer to the @SRH run function (5.1.37) for definitions. The only exception is that the RID must be specified when using the @SRU run function.

Example:

```
@SRU,102,B,4,,99 '' F2-4 ,8001/R,8012 V1I4,V2I5
```

Searches for lines in which the four characters from the first column of the second field of the RID 4 report are in the range 8001 to 8012. Branch to label 99 if no lines are found. The number of lines found is entered in variable V1, and the number of lines searched is put in V2.

```
@DEL
```

Delete the found lines from the original report.

5.1.39. @SUB (Subtotal) Run Function

The @SUB run function calculates the subtotal of a field or fields based on a change in the specified key. Follow the @SUB function directly with an output line, since one line of output is created for each subtotal.

When there is no output line, if the line directly after the @SUB is not an output line, or if the output line does not contain the variable defined in the @SUB function, the run ends in error.

Format:

```
@SUB,mode,type,rid,[label] options characters
  [line-type],parameters variables
```

where:

`mode`
Specifies the mode of the report.

`type`
Specifies the type of the report.

`rid`
Specifies the RID number of the report.

`label`
Specifies the destination label if the line-type (tab or *) to be processed is not in the specified report.

`options`
Refer to options in Appendix B.

`characters`
Specifies the fields to subtotal.

`line-type`
Specifies the line type (tab or *) to be processed.

`parameters`

Specifies the processing control parameters. The following three control parameters can be used for this parameter:

S Specifies the key field. The length must be within 24 characters.

\+ Specifies the fields to be subtotaled with a maximum of 16.

M Specifies that the designated field in the last line of the key group be entered in the variable. The M is optional and a maximum of 16 characters can be specified.

`variables`

Specifies the variables that contain the processing results.

When the E option is specified, the first variable contains the number of entries of the specified line-type of each subtotal group. When there is no specification of option E, this variable is the same as the second variable.

The second and subsequent variables contain data corresponding to the symbol specified by the parameter. Refer to the parameter definitions of S, +, and M.

NOTES:

1. *Other run functions are not allowed on the same line after the @SUB run function.*
2. *The line after the @SUB run function must be an output line that contains the variables defined in the @SUB run function.*

Example:

The report to be processed by the run example is 102,D,2. The run report is in mode 102, type B, and the result produced by the run is in mode 102, type B, RID (–0).

Report contents:

```
LINE> 1      FMT>   RL>        SHFT>       HLD CHR>    HLD LN>    PSWD>           >
.DATE  83/07/13  12:24:15  TYPE=D  RID=002  83/07/13  JDOE          <   20 LINES>
. <<<  CORPORATE ORDER STATUS  >>>
*ST.ORDER . PRODUCT .ODR.CUST.  UNIT  .EXTENDED.REQ'D .SALE.                .
*CD.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REP .    CUSTOMER     .
*==.======.=========.===.====.========.========.======.====.================.==
 OR 99951S GREENBOX9   2 AMCO    26775    53550 750312 DJR  AMERIAN OIL CO.   73
 OR 99951S GREENBOX7   1 AMCO    26075    26075 750312 DJR  AMERIAN OIL CO.   73
 OR 99951S BLACKBOX9   1 AMCO    25025    25025 750312 DJR  AMERIAN OIL CO.   73
 OR 96652S GREENBOX4   2 ARCO    25025    50050 750412 LSJ  ARGENTINE CORP.   23
 OR 96652S BLACKBOX5   1 ARCO    24325    24325 750412 LSJ  ARGENTINE CORP.   23
 OR 96652S BLACKBOX4   1 ARCO    24150    24150 750412 LSJ  ARGENTINE CORP.   23
 OR 99753S GREENBOX5   1 DICO    25375    25375 750312 LSJ  DIGITAL CORP     17
```

Define the run:

```
@BRK
 -----------------     -------  ------  ---------------
    CUSTOMER           ENTRIES  ORDERS  EXTENDED RETAIL
 -----------------     -------  ------  ---------------
@SUB,102,D,2 E F4,F7,F10 ,+,+,S V1I3,V2I3,V3I8,V4S17
 V4      V1    V2    V3
@GTO END
                      ..... END REPORT .....
```

where:

@BRK
: Clears the output area.

@SUB,102,D,2 E F4,F7,F10 ,+,+,S V2I3,V2I3,V3I8,V4S17

where:

102,D,2
: Specifies the report to be subtotaled.

E
: Specifies the option that produces a count of the number of entries per line-type, per subtotal group.

F4,F7,F10
: Specifies the fields to be processed.

  F4 Is the fourth field in the report (ORD QTY).

  F7 Is the seventh field in the report (EXTENDED RETAIL).

  F10 Is the tenth field in the report (CUSTOMER).

,+,+,S
: Specifies the control parameters to accumulate the subtotals for F4 and F7 by subkey F10.

V1I3,V2I3,V3I8,V4S17
: Specifies the variables that contain the results:

  V1 Contains the number of entries.

  V2 Contains the subtotal for F4 (ORD QTY).

  V3 Contains the subtotal for F7 (EXTENDED RETAIL).

  V4 Contains the subtotal key (CUSTOMER).

V4 V1 V2 V3
Writes the results to the output area.

@GTO END
Displays the results and terminate the run.

Results produced by the run:

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD> ''RESULT''>
.DATE  83/07/14  15:33:05  TYPE=B  RID=     83/07/14  MAPR          <   11 LINES>
 ----------------    -------  ------  --------------
    CUSTOMER          ENTRIES  ORDERS  EXTENDED RETAIL
 ----------------    -------  ------  --------------
 AMERIAN OIL CO.          3       4       104650
 ARGENTINE CORP.          3       4        98525
 DIGITAL CORP             1       1        25375
 FED SYSTEMS CORP         4       4        75250
 INTERNATIONAL CO         1       1        24150
 UNION STEEL/SULFR        2       2        51800
                    ..... END REPORT .....
```

5.1.40. @TOT (Totalize) Run Function

The @TOT run function creates results by performing arithmetic, move and fill operations on fields within a report.

Format:

```
@TOT,mode,type,rid,[label] options characters
  [line-type],parameters [variables]
```

where:

mode
Specifies the mode of the report to be totalized.

type
Specifies the type of the report to be totalized.

rid
Specifies the report number of the report to be totalized.

label
Specifies the destination label when no lines are found.

options
Specifies options (Appendix B).

characters
Specifies the fields, partial fields, or character positions to be processed.

`line-type`
Specifies the line type (tab type) to be processed.

`parameters`
Specifies parameters corresponding to the field to be processed. These include: +, -, *, /, =, A, M, S, C

where:

A Specifies average.

M Specifies move.

S Specifies subtotal.

C Specifies cumulation.

`variables`
Specify the variables to hold the results. Variables are effective only for vertical totalizations and average calculations. They are ignored for other operations (horizontal calculations, move data, etc). The output generated for the vertical and average operations can, depending on the inclusion or omission of variables, be of two formats.

Format 1 – Inclusion of variables:

The output consists only of totals contained in the variables.

Format 2 – Omission of variables:

The output consists of a result report with the totals, as part of the report, included at the end.

NOTES:

1. *When specifying the E option, the first variable contains the number of data lines (the number of entities) processed.*

2. *Vertical total or average calculation results are entered in the second and subsequent variables.*

3. *When no E option is specified, all variables contain the vertical total or average calculation results.*

4. *The maximum number of effective digits of the calculation result is 15. Digits after the fifteenth are truncated.*

Examples:

```
@TOT,102,C,1,99 '' F4,F6,F7 ,+,*,=
```

Indicates horizontal calculation, which multiplies the numeric value of the fourth field by the numeric value of the sixth field and enters the product in the seventh field.

```
@TOT,102,C,1,99 E F4,F5,F6 ,+,+,+ V1I4,V2I8,V3I8,V4I8
```

Calculates the vertical total data in the fourth, fifth, and sixth fields and enters the results in V2, V3, and V4. Since there is an E option specified, the number of entries (data lines) are entered in V1. A result report is not created.

```
V1  V2  V3  V4
```

Writes the totals to the output area.

```
@DSP,100,D,-0
```

Displays the totals.

5.1.41. @ULK (Unlock) Run Function

The @ULK run function releases update control of the report that was locked by the @LOK run function.

Format:

```
@ULK
```

5.1.42. @UPD (Update) Run Function

The @UPD run function updates the lines in the original report corresponding to the lines in the update results created by the @MAU or @SRU run functions.

The report updated is the receiving report of the @MAU or the original report of the @SRU run function.

Format:

```
@UPD
```

Example:

```
@MAU,102,C,1,102,B,2 M F1,F3 ,1,A F4,F6 ,1,A
```

MATCH (UPDATE) the first field in the issuing report (102,C,1) against the fourth field in the receiving report (102,B,2). When a match occurs, move the matching line of the receiving report to the result. Then move the third field of the issuing report to the sixth field of the result. The M option specifies that only matching lines are to be included in the result.

@UPD

Replace the matching lines in the receiving report with the corresponding lines from the update result.

5.1.43. @WRL (Write Line) Run Function

The @WRL run function writes data in up to 23 lines of a report. It uses either literal values or input from variables. When executing this function, it is necessary to lock this report in advance with the @LOK run function.

Format:

```
@WRL,mode,type,rid,start-line,[identification],[password]
 characters [line-type],parameters
```

where:

mode
Specifies the mode of the report to be written to.

type
Specifies the type of the report to be written to.

rid
Specifies the report number of the report to be written to.

start-line
Specifies the starting line number for writing.

identification
When Y is specified, do not update the time and user-id.

password
When specified, assigns or changes a report password.

characters
Specifies the fields, partial fields, or character columns in which to write the data.

line-type
Specifies the line type designator to write to. The @WRL run function converts all updated lines to the line type specified in this subfield regardless of their original type. The default equals the tab type.

parameters
Specifies the data to write in the specified fields.

NOTES:

1. *When writing multiple lines, delimit the line type and parameters of each line with slashes.*

2. *If the specified line type and the original line type differ, the data content is not guaranteed.*

Example 1:

Original report:

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>          >
.DATE  83/07/05  10:47:50  TYPE=B  RID=003  83/06/29  JDOE        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 IP 741224 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 IP 741225 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 IP 741219 LS BLACKBOX2 637071        84353 INTR 741218 741219
 OR 750110 LS BLACKBOX4               94754 ARCO
 SC 750110 LS BLACKBOX5 675281        97441 FEDS 750131
```

This run writes one line to the original report:

```
@CHG V11A2  A1
@CHG V12A6  810101
@LOK,102,B,3
@WRL,102,B,3,6 F1,F2 [,V11,V12
@ULK
@GTO END
                    ..... END REPORT .....
```

Original report after the run has updated the first data line:

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>          >
.DATE  83/07/13  13:04:11  TYPE=B  RID=003  83/07/13  MAPR        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.=========.======.======.=====.====.======.======.======.=====.===.
 A1 810101 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 IP 741225 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 IP 741219 LS BLACKBOX2 637071        84353 INTR 741218 741219
 OR 750110 LS BLACKBOX4               94754 ARCO
 SC 750110 LS BLACKBOX5 675281        97441 FEDS 750131
```

Example 2:

This run writes multiple lines to the original report.

```
@CHG V11A2  A1    CHG V21A2  A2    CHG V31A2  A3
@CHG V12A6  810101    CHG V22A6  820101    CHG V32A6  830101
@LOK,102,B,3
@WRL,102,B,3,6 F1,F2 [,V11,V12/[,V21,V22/[,V31,V32
@ULK
@GTO END
                      ..... END REPORT .....
```

Original report after the run has updated the first three data lines:

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>           >
.DATE  83/07/13  13:25:17  TYPE=B  RID=003  83/07/13  MAPR        <   48 LINES>
. <<<  CORPORATE PRODUCTION STATUS  >>>
*ST.STATUS.BY. PRODUCT .SERIAL.PRODUC.ORDER.CUST.PRODUC.PRODUC. SHIP .SHIP .SPC.
*CD. DATE .IN.  TYPE   .NUMBER. COST .NUMBR.CODE. PLAN .ACTUAL. DATE .ORDER.COD.
*==.======.==.========.======.======.=====.====.======.======.======.=====.===.
 A1 810101 LS BLACKBOX1 436767        84389 AMCO 741223 741224
 A2 820101 LS BLACKBOX1 436768        84390 AMCO 741223 741225
 A3 830101 LS BLACKBOX2 637071        84353 INTR 741218 741219
 OR 750110 LS BLACKBOX4               94754 ARCO
 SC 750110 LS BLACKBOX5 675281        97441 FEDS 750131
```

5.1.44. @XIT (Sign Off) Run Function

The @XIT run function ends the run being executed and puts the workstation in the post sign-off state. An idle logo is then displayed on the workstation screen.

Format:

```
@XIT
```

6. Run Creation Example

6.1. GENERAL

The example provided in this section calculates and displays grand totals of the quantity and value of the orders received and subtotals by customer. You have the option of printing the results on an auxiliary (AUX) printer. Two reports are included in this process:

- Corporate Factors Base , report 102,C,1

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>           >
.DATE  83/06/29  13:07:49  TYPE=C  RID=001  83/06/29  JDOE        <   24 LINES>
. <<<  CORPORATE FACTORS BASE  >>>
* PRODUCT . SUB .PRODUC. WHOLE . RETAIL . SALES .SPACE.  DEMO  .              .
*  TYPE    . KEY . COST . SALE$ .  $$$$  .COMMISS. REQ .QUANTITY. DEMO RESULTS  .
*=========.=====.======.=======.=======.=======.=====.========.===============.
 BLACKBOX1     A  13500   16875    23625 2362.50   100        1
 BLACKBOX2     A  13600   17000    23800 2380.00   110        2
 BLACKBOX3     A  13700   17125    23975 2397.50   120        4
 BLACKBOX4     B  13800   17250    24150 2415.00   130       10
 BLACKBOX5     B  13900   17375    24325 2432.50   140       50
 BLACKBOX6     C  14000   17500    24500 2450.00   150      100
 BLACKBOX7     C  14100   17625    24675 2467.50   160       10
 BLACKBOX8     D  14200   17750    24850 2485.00   170       20
 BLACKBOX9     D  14300   17875    25025 2502.50   180       40
 GREENBOX1     E  13700   17125    23975 2397.50   200       80
 GREENBOX2     E  13900   17375    24325 2432.50   210      160
 GREENBOX3     E  14100   17625    24675 2467.50   220        5
 GREENBOX4     F  14300   17875    25025 2502.50   230       15
 GREENBOX5     G  14500   18125    25375 2537.50   240       25
 GREENBOX6     H  14700   18375    25725 2572.50   250        1
 GREENBOX7     I  14900   18625    26075 2607.50   260        2
 GREENBOX8     J  15100   18875    26425 2642.50   270        3
 GREENBOX9     K  15300   19125    26775 2677.50   280        4
                    ..... END REPORT .....
```

- Corporate Order Status , report 102,D,1

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD>          >
.DATE  83/07/05  09:29:51  TYPE=D  RID=001  83/06/29  JDOE        <   20 LINES>
. <<<  CORPORATE ORDER STATUS  >>>
*ST.ORDER . PRODUCT .ODR.CUST.  UNIT  .EXTENDED.REQ'D .SALE.                .  *                .          .      .      .
*CD.NUMBER.  TYPE   .QTY.CODE. RETAIL . RETAIL .DELIVR.REP .    CUSTOMER    .  * ADDRESS       .    CITY  .STATE. ZIP  .REMARK.
*==.======.=========.===.====.========.========.======.====.================.==*=============.==========.=====.=====.======.
 OR 99951S GREENBOX9   2 AMCO                  750312 DJR  AMERIAN OIL CO.  73  00 CENTRAL AV NEW ORLEANS      LA 64301
 OR 99951S GREENBOX7   1 AMCO                  750312 DJR  AMERIAN OIL CO.  73  00 CENTRAL AV NEW ORLEANS      LA 64301
 OR 99951S BLACKBOX9   1 AMCO                  750312 DJR  AMERIAN OIL CO.  73  00 CENTRAL AV NEW ORLEANS      LA 64301
 OR 96652S GREENBOX4   2 ARCO                  750412 LSJ  ARGENTINE CORP.  23  00 5TH AVE     NEW YORK         NY 33021
 OR 96652S BLACKBOX5   1 ARCO                  750412 LSJ  ARGENTINE CORP.  23  00 5TH AVE     NEW YORK         NY 33021
 OR 96652S BLACKBOX4   1 ARCO                  750412 LSJ  ARGENTINE CORP.  23  00 5TH AVE     NEW YORK         NY 33021
 OR 99753S GREENBOX5   1 DICO                  750312 LSJ  DIGITAL CORP     17  82 NORTH ST    NEW YORK         NY 54002
 OR 94525S GREENBOX8   1 FEDS                  750312 PLR  FED SYSTEMS CORP 15  66 COLUMBIA    WASHINGTON       DC 20001
 OR 96751S GREENBOX1   1 FEDS                  750312 PLR  FED SYSTEMS CORP 15  66 COLUMBIA    WASHINGTON       DC 20001
 OR 99842S BLACKBOX8   1 FEDS                  750312 PLR  FED SYSTEMS CORP 15  66 COLUMBIA    WASHINGTON       DC 20001
 OR 99842S BLACKBOX0   1 FEDS                  750312 PLR  FED SYSTEMS CORP 15  66 COLUMBIA    WASHINGTON       DC 20001
 OR 99752S BLACKBOX4   1 INTR                  750312 LTR  INTERNATIONAL CO 33  01 SUMMIT AV   CHICAGO         ILL 65320
 OR 98782S BLACKBOX9   1 USSC                  750312 SSF  UNION STEEL/SULFR 54  30 ALCAN AVE   SEATTLE        WASH 73001
 OR 96755S GREENBOX9   1 USSC                  750312 SSF  UNION STEEL/SULFR 54  30 ALCAN AVE   SEATTLE        WASH 73001
                    ..... END REPORT .....                                                    ..... END REPORT .....
```

6.2. PROCESSING FLOW

Figure 6–1 shows the processing flow of the run.

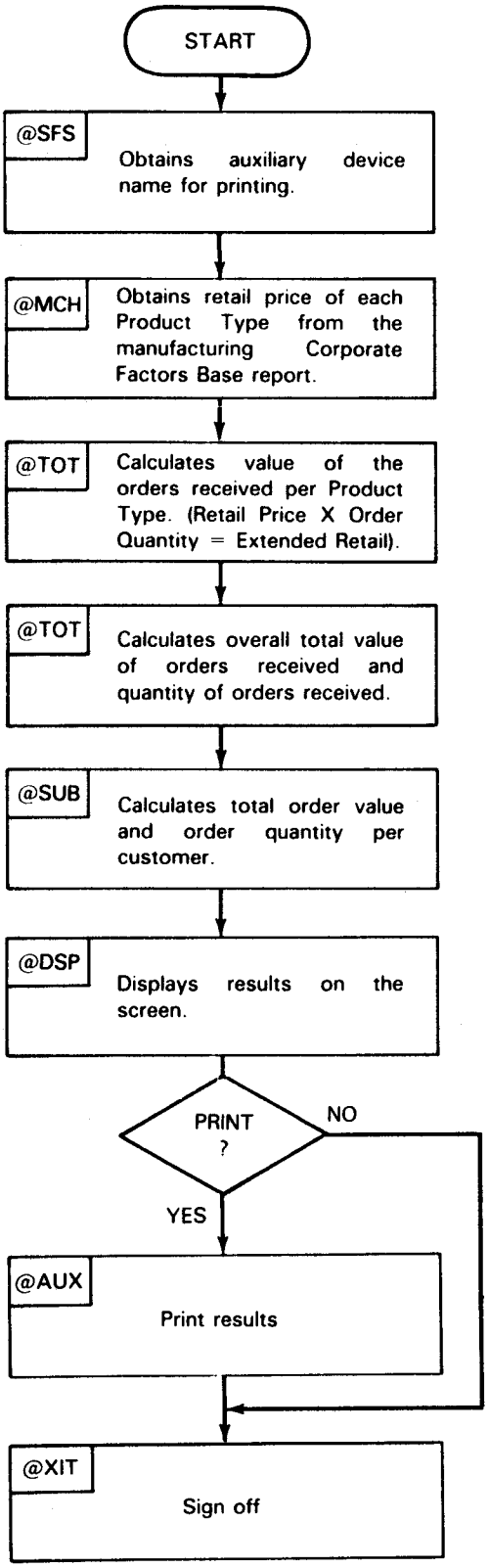


Figure 6–1. Processing Flow

6.3. RUN DETAILS

The run details are given in Figure 6–2.

```
LINE> 1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN> 3  PSWD>           >
*COLUMN NUMBER
         1111111111222222222233333333334444444444555555555566666666667777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890
.DATE  83/07/14  13:51:38  TYPE=G  RID=018  83/07/13  JDOE         <   39 LINES>
.MAPPER80 RUN FUNCTION : RUN-NAME = RUNSTA      : EXAMPLE (ORDER STATUS SUMMARY)
*================================================================================
@BRK
@SFS U100  /V1A3                                          . GET AUX NUMBER
@CHG V2H6 DATE1$                                          . GET DATE
@CHG V3A2 V2(1-2)  CHG V4A2 V2(3-2)  CHG V5A2 V2(5-2)     . EDIT DATE
@CHG V11H8 TIME$                                          . GET TIME

         ---------------------------------
                 ORDER STATUS SUMMARY
           DATE: V3/V4/V5  TIME: V11
         ---------------------------------

@MCH,102,C,1,102,D,1 '' F1,F5 ,1,A F3,F6 ,1,A             . GENERATE RESULT FROM 1D.
@.                                                        . INCLUDE RETAIL PRICE
@.                                                        . FROM 1C
@TOT,102,D,-0  ''  F4,F6,F7  ,*,+,=                       . MULTIPLY RETAIL PRICE
@.                                                        . BY NUMBER OF ORDERS.
@RNM -1                                                   . SAVE RESULT
@TOT,102,D,-1  ''  F4,F7    ,+,+  V13I3,V14I6             . TOTALIZE NUMBER OF
@.                                                        . ORDERS AND RETAIL SALES.

     CUSTOMER          ORDER QTY      RETAIL SALES
 ----------------     ---------      -------------
@SUB,102,D,-1 '' F4,F7,F10 ,+,+,S  V15I3,V16I8,V17S17     . SUBTOTAL BY CUSTOMER
 V17       V15          V16
 ----------------     ---------      -------------
     TOTALS           V13 ORDERS     $V14 SALES
 ----------------     ---------      -------------
@BRK
@DSP,102,G,-0                                             . DISPLAY THE RESULT
@IF V1 = ' ' GTO 99                                       . TEST IF RESULT IS TO
@.                                                        . BE PRINTED ON AUX.
@AUX,102,G,-0,V1,,Y                                       . PRINT RESULT ON AUX V1
@99:XIT                                                   . SIGN OFF
                        ..... END REPORT .....
```

Figure 6–2. Run Details

6.4. RUN EXECUTION

Run execution first requires starting the run. Refer to 4.2.1 through 4.2.3. This consists of entering the @RUN run function followed by the run name:

```
@RUNSTA.
```

When the run starts, the following screen format services screen appears:

```
 <<< ORDER STATUS SUMMARY >>>

IF YOU WANT TO PRINT, PLEASE ENTER THE AUXILIARY DEVICE ID: (___)
```

To print the displayed result on the auxiliary (AUX) printer, key in the auxiliary printer name (format=Pxx) and transmit. If the display is not to be printed, leave the field blank and transmit. After transmitting, the run resumes and performs the required calculations and displays the result on the screen.

```
LINE >1     FMT>   RL>       SHFT>      HLD CHR>    HLD LN>    PSWD> ''RESULT''>
.DATE  83/07/14  12:54:20  TYPE=G  RID=     83/07/14  JDOE          <   19 LINES>

          ----------------------------------
                ORDER STATUS SUMMARY
            DATE: 83/07/14  TIME: 12:53:55
          ----------------------------------

     CUSTOMER          ORDER QTY      RETAIL SALES
 ----------------      ---------      -------------
 AMERIAN OIL CO.           4             104650
 ARGENTINE CORP.           4              98525
 DIGITAL CORP              1              25375
 FED SYSTEMS CORP          4              75250
 INTERNATIONAL CO          1              24150
 UNION STEEL/SULFR         2              51800
 ----------------      ---------      -------------
     TOTALS            16 ORDERS     $379750 SALES
 ----------------      ---------      -------------
                     ..... END REPORT .....
```

Key in RSM and transmit. The run resumes and, based on the response entered in the SFS screen display, decides if the results are to be printed on the auxiliary printer.

If the SFS screen response contained an auxiliary name, the run prints the result on the AUX printer, then terminates and signs off.

If the SFS screen response was blank, the run terminates and signs off.

Appendix A. Run Function Summary

The run functions are shown in A.1 through A.7.

A.1. REPORT UPDATE FUNCTIONS

@ADD Creates a result by appending one report to another.

@ADR Creates a new report.

@BRK Converts data in output area to a result, erases the contents of the output area, and reopens it.

@DLR Deletes reports.

@DUP Duplicates reports.

@REP Replaces one report with the contents of another report.

@RNM Renames report or result (–0) to a temporary –1 to –4 result name.

A.2. LINE UPDATE FUNCTIONS

@DEL Deletes lines of the original report corresponding to the update result lines.

@LIN Obtains line number of the last line of the report.

@LNX Duplicates lines in the report.

@LN+ Adds lines to the report.

@LN– Deletes lines from the report.

@RDC Reads multiple lines in the report continuously and writes that data in the output area.

@RDL Reads one line from a report.

@RLN Reads the next line of a report.

@UPD Updates lines of the original report corresponding to the lines of the update result.

@WRL Writes data in a report.

A.3. QUERY FUNCTIONS

@BFN Finds data in a report by means of the binary division method.

@FND Finds specified data in the report.

@LCH Produces a result by locating a certain character string in the report and replacing it with another character string.

@MAU Matches two reports and produces a result that can be used to update the receiving report.

@MCH Matches two reports and produces a result.

@SOR Sorts data in the report. Produces a result.

@SRH Finds all specified data in the report and produces a result.

@SRU Finds all specified data in the report and produces a result that can be used to update the original report.

A.4. CALCULATION FUNCTIONS

@CHG Changes contents of a variable.

@SUB Prepares subtotal of data in the report.

@TOT Performs arithmetic operations.

A.5. WORKSTATION DISPLAY FUNCTIONS

@DSP Displays the report at the workstation.

@SFS Displays the format screen prepared by screen format services at the workstation.

A.6. RUN END FUNCTIONS

@GTO END Completes the run and displays the contents of the output area on the screen.

@REL Ends run (goes into SIGN ON state, user logo displayed).

@XIT Ends run (goes into SIGN OFF state, idle logo displayed).

A.7. OTHER FUNCTIONS

@AUX Prints report or result on the named auxiliary printer.

@COP Prints report or result on the COP printer atttached to the workstation.

@ESR Returns from a subroutine.

@GTO Performs branching.

@IF Compares and assesses variable data.

@IND Creates report index in the specified type.

@INS Inserts data into variable.

@LOK Prevents other users from updating the same report that the current run is updating.

@PRT Prints report on the system printer.

@RSR Executes a subroutine.

@RUN Ends run being executed and starts execution of a different run.

@ULK Releases lock on report.

Appendix B. Options Table

Options	Function	Run Instruction							
		@BFN	@FND	@LCH	@MAU/@MCH	@SRH	@SRU	@SUB	@TOT
A	Processes all line types.		X	X		X	X		
D	Deletes search or match lines from the result.				X	X	X		
Dn	Rounds down answer to nearest n.							X	X
E	Counts number of entries.							X	X
F	Processes the first column of the mask parameter as part of the character string.			X					
F	Leaves original data in the receiving field of the unmatched lines.				X				
H	Displays the header line only of the initial report of the multiple reports searched.					X			
I	Deletes the index and item name of the total, subtotal, and average.								X
M	Substitutes line type when a substitution occurs in the line.			X					
M	Displays the matched lines only.				X				
N	Displays data other than the specified parameter.					X	X		
N	Displays the unmatched lines only.				X				
O	Displays total, average, and subtotal lines only.								X

Options	Function	Run Instruction							
		@BFN	@FND	@LCH	@MAU/@MCH	@SRH	@SRU	@SUB	@TOT
P	Indicates that the key field has been sorted.				X				
Q	Stops search after the first find.	X							
Rn	Rounds answer to nearest n.							X	X
Rm-n	Searches multiple reports.		X			X			
Sn	Specifies starting line number for scanning.			X					
Tx	Sets transparent character to x.			X					
T	Displays vertical summations, averages, and subtotals as tab line – lines when used in combination with the V option.								X
Un	Rounds up answer to nearest n.							X	X
V	Displays total, average, and subtotal under each item header.								X
@	Blank fields	X	X			X	X		
/	Process slashes (/) as a part of the character string.	X	X			X	X		

Appendix C. Correspondence Between Run Functions and Manual Functions

Run Functions	Corresponding Manual Functions
@ADD	ADON
@ADR	AR
@AUX	AUX
@BFN	BF
@COP	COP
@DEL	DEL
@DLR	DR
@DSP	D
@DUP	XR
@FND	F
@IND	I
@LCH	CHG
@LNX	▷]gXn (Duplicate line.)
@LN+	▷]g+n (Add line.)
@LN–	▷]g– (Delete line.)

Run Functions	Corresponding Manual Functions
@MAU	MAU
@MCH	MA
@PRT	PR
@REL	^
@REP	REP
@RUN	RUN
@SOR	SORT
@SRH	S
@SRU	SU
@TOT	TOT
@UPD	UPD
@WRL	▷ *** *** / (SOE update)
@XIT	X

Index

CUT

SPERRY

USER COMMENT SHEET

We will use your comments to improve subsequent editions.

NOTE: Please do not use this form as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

From:

_____

(Name of User)

_____

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

CUT

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY CORPORATION

ATTN.: SOFTWARE SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

FOLD