

Document Composition Facility

SCRIPT/VS Text Programmer's Guide

Release 4.0



Document Composition Facility

SH35-0069-06

SCRIPT/VS Text Programmer's Guide

Release 4.0

Seventh Edition (January 1991)

This edition applies to Release 4.0 of the Document Composition Facility (DCF) licensed program, Program Number 5748-XX9, and to all subsequent releases and modifications until otherwise indicated in new editions or Technical Newsletters. Be sure to use the correct edition for the level of the product.

Technical changes in this edition are marked by vertical bars in the left margin.

Consult the latest edition of the applicable IBM system bibliography for current information on this product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation
Information Development
Department 588
PO BOX 1900
Boulder, Colorado 80301-9191

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1984, 1991. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Introduction to the Text Programmer's Guide	3
Printer References	4
Major Divisions of This Publication	4
Chapter 2. Overview of SCRIPT/VS	9
Operating Environments	9
Logical Output Devices and Output Destinations	9
SCRIPT/VS Utility Files	10
Communicating with SCRIPT/VS	10
SCRIPT/VS Functions	11
Formatting Functions	11
SCRIPT/VS Programming Facilities	11
General Document-Handling Functions	12
Calling the SCRIPT/VS Processor	13
Interactive Environments	13
Batch Environments	13
Using SCRIPT/VS as a Subroutine	13
Using SCRIPT/VS as a Preprocessor	13
SCRIPT/VS as a Processor for RFTDCA	13
Chapter 3. Defining the Formatting Environment	17
Formatting Environment Parameters	17
Running Heading and Footing Environments	17
Keep, Float, Footnote, Table, and Named Area Environments	17
Saving and Restoring the Current Formatting Environment	18
Named Environments	18
Chapter 4. Processing Symbols	23
Symbols for SCRIPT/VS Control Values	23
&\$ASCI Special Symbol	24
&\$DCF Special Symbol	24
&\$DDUT Special Symbol	24
&\$EGML Special Symbol	24
&\$ENV Special Symbol	25
&\$ENVARRAY Special Symbol	25
&\$GML Special Symbol	25
&\$IXLG Special Symbol	25
&\$LC Special Symbol	26
&\$LST Special Symbol	26
&\$PASS Special Symbol	26
&\$PRT Special Symbol	26
&\$PSNO Special Symbol	27
&\$RET Special Symbol	27
&\$RH and &\$RF Special Symbols	27
&\$SEPM Special Symbol	28
&\$SVLG Special Symbol	28
&\$TAG Special Symbol	28
&\$TAGD Special Symbol	28
&\$VR Special Symbol	28
Passing Parameters to Input Files	29
Setting Symbols with the SCRIPT Command	29

Symbols Set When a File Is Imbedded or Appended	29
Symbols Set When a Macro Is Processed	30
Setting a Symbol to the Current Page Number	31
Symbols for Arrays of Values	31
Controlling the Array Elements	32
Extended Symbol Processing	35
Defining Text Variables	36
Producing Special Characters	36
Overriding Delimiter Characters	37
Using Defined Variables to Change Fonts	38
Attributes of a Symbol's Value	39
Chapter 5. Processing Logical Statements	43
The .IF Control Word Family	43
Alternative Processing	44
Bypassing Part of an Input File	45
The SYSPAGE and SYSOUT Comparands	46
Special Techniques for Conditional Processing	46
Conditional Sections	47
Logical Processing With Symbols	49
Chapter 6. Processing Macros	53
When to Use Macros	53
How to Define a Macro	54
How Values Are Substituted for Symbols within a Macro Definition	56
Conditional Macro Processing	57
Macro-Naming Conventions	57
Local Symbols for Macros	57
Processing Local Variables	59
Terminating a Macro	59
Redefining SCRIPT/VS Control Words	60
Using Symbols and Macros as Associative Memory	61
Redefining SCRIPT/VS Formatting Conventions	63
Processing Input Lines That Begin with a Blank or a Tab	63
Specifying a Macro Library	64
LIB: Specify Symbol and Macro Libraries	64
Creating SCRIPT/VS Macro Libraries	66
In a CMS Environment	66
In a TSO Environment	66
In an ATMS-III Environment	67
In a DLF Environment	68
Chapter 7. Processing GML	71
GML Markup Syntax	72
Changing the GML Delimiters	73
SCRIPT/VS Processing of GML	74
Automatic GML Processing	74
Attribute Scanning Rules	75
GML Tag-to-APF Mapping	79
Creating Your Own GML Tag	80
Chapter 8. Verifying Spelling	85
Spelling Verification	85
Spelling Fallibility	88
The SCRIPT/VS Dictionaries	89

Building a User Dictionary	91
Building an Addenda Dictionary	91
TLIB: Specify Spelling Checking and Hyphenation Libraries	92
Searching a SCRIPT/VS Dictionary	93
Stem Processing	93
English Prefixes and Suffixes	94
French National and Canadian French Prefixes and Suffixes	95
Dutch Prefixes and Suffixes	96
Italian Prefixes and Suffixes	97
German Prefixes	99
Spanish Prefixes	101
Chapter 9. Performance Considerations	105
Storage Usage	105
SCRIPT Command Options	105
FPASSES(n) Option	106
INDEX Option	106
SEPMASR Option	106
SPELLCHK Option	106
TWOPASS Option	107
Mapped Page Segments	107
Changing the NOMAP Default	107
Removing Extra Languages from the GML Starter Set	107
SCRIPT/VS in the ATMS-III Environment	108
Tuning ATMS-III for SCRIPT/VS	108
Chapter 10. Tailoring SCRIPT/VS for Your Installation	111
Procedure for Tailoring SCRIPT/VS	111
Changing the Resident Dictionary	112
Source Files Provided with SCRIPT/VS	112
DSMCSMDM: Message Definitions	113
DSMTSMDM: Additional Message Definitions for TSO	115
DSMCSPDB: Phrase Definitions	116
DSMeeUOT: Environment-Dependent Options Tables	119
DSMLPBCS: Box Character Sets	130
DSMLPLDT: Logical Device Definitions	131
DSMLPPDT: Physical Device Tables	136
DSMTFFIB: Line Printer Font Information	140
DSMTRNLT: National Language Character Definitions	144
Index Sort Sequence Macros	147
Replacing the IBM-Supplied Dictionaries	150
Required Fonts for Page Printers	151
Required Fonts for PostScript Devices	153
Chapter 11. Creating and Maintaining User Dictionaries	157
Dictionary Maintenance	157
Sample Procedure	160
MVS PROC Defaults	161
Table of STEPS, DDNAMES, and DSNs for the DM PROC	162
Chapter 12. National Use Considerations	169
Language Attribute on the :GDOC Tag	169
Index Sort Sequence	169
Spelling Verification and Hyphenation Dictionaries	170
Accented Characters	171

Message Translation	171
Default Language	172
Default Font for Page Mode Devices	172
Katakana	173
Uppercase String Translation	173
Other Considerations	173
PostScript Code Pages	173
Chapter 13. Index Sort Sequence	177
Sort Sequences	177
Modifying the Index Sort Sequences	181
The 1-to-2 Table	181
The 1-to-1 Table	181
Assigning Sort Values	182
Header Characters	183
Character Code Points	184
The Index Sort Modules	184
Chapter 14. Using Separation Masters	189
Creating Separation Masters	189
Steps to Creating Separation Masters	191
Using the SEPMASTR Command Option	191
Using the .SM Control Word	192
Using SEPMASTR Keyword with Control Words	192
.DF Control Word	192
.DR Control Word	193
.OI Control Word	194
.PO Control Word	194
.SD Control Word	194
.SI Control Word	195
Default Master	195
Producing Separation Masters	196
Creating Multiple Color Graphics	200
Modifying the GML Profiles	201
Fonts	201
Rules and Tables	202
Appendix A. Using SCRIPT/VS with Other IBM Programs	207
Producing Input for STAIRS/VS	207
Specifying STAIRS/VS Output	207
Restrictions Imposed on Formatted Output	207
STAIRS/VS Paragraph Identification	208
Using SCRIPT/VS as a Postprocessor	209
Using SCRIPT/VS as a Preprocessor	210
Developing Preprocessor APFs and Profiles	210
Redefining Symbols	211
Handling Directly Entered Control Words	211
Preparing for Processing	212
Appendix B. Compatibility with Earlier Releases of SCRIPT/VS	215
Release 4.0 of the Document Composition Facility	215
Release 3.2 of the Document Composition Facility	217
Release 3.1 of the Document Composition Facility	217
Release 3.0 of the Document Composition Facility	218
Release 3.0 Compatibility with Release 2 of SCRIPT/VS	219

Changes to the SCRIPT Command	219
Changes to SCRIPT Control Words	220
Running Headings and Footings in SCRIPT/VS Release 3	223
SCRIPT/VS Macro Libraries	229
Tabs	229
The SCRIPT/370 Dictionary	229
Unsupported Control Words	229
Changes to SCRIPT Command Options	231
Changes to SCRIPT/VS Control Words	234
Compatibility with TSO/FORMAT	242
Creating a TSO/FORMAT-Compatible Environment	243
Appendix C. IBM 3800 Printing Subsystem Model 1 Fonts Provided with SCRIPT/VS	247
The SCRIPT/VS Fonts	247
SCRIPT/VS Font Modification	249
Creating a World Trade Font	250
Fonts Available for Use with SCRIPT/VS	252
Appendix D. The DCF Font Library Index Program	261
Operating Environments	261
CMS	262
Using the Font Library Index Program with TSO	263
Concatenation Considerations with FLIP	263
MVS	264
VSE	265
Font Library Index Program Return Codes	266
Font Library Index Program Report	266
Appendix E. PostScript Information	277
PostScript Font Support	277
Adobe Font Metrics Files	277
Adobe Font Metrics Files	278
PostScript Code Pages	279
The PostScript DCFINDEX File	281
Configuring Your Printer for 8-Bit ASCII	286
PostScript Output Structuring Conventions	286
PostScript Command Abbreviations	287
Other PostScript Considerations	287
Appendix F. DCF Programming Interface for CICS	291
Interface Description	292
General Sequence of Events	292
Detailed Description	292
Other Considerations	303
Terminal Output Considerations	303
Device Type Determination	303
CICS Services Used by SCRIPT/VS	304
CICS Generation	304
Generalized Markup Language Macros	304
Font Library Support	305
Page Segment Library Support	305
Additional Interface Control Block Information	305
Interface Control Block	306
Command Options Block	309

Appendix G. Algorithmic Hyphenation Interface in DCF	313
Introduction	313
Hyphenation Processing in SCRIPT/VS	313
Hyphenation Algorithms in SCRIPT/VS	314
Algorithmic Hyphenation Routine Interface	315
Expanded Interface Capabilities	315
Register Conventions	315
The Parameter List	316
Routine Entry and Exit Protocol	317
Installation Instructions	317
The CMS Environment	317
The TSO Environment	318
 Appendix H. Related Publications and Products	 323
Related Products	323
Publication Library Guide for the Document Composition Facility	324
DCF Publications	325
Optional Features	325
Related Publications	326
 Glossary	 327
Source Identifiers	327
References	327
 Index	 339

Figures

1.	Processing Documents with GML	75
2.	Example of a Font Width Table	143
3.	Sample Insurance Form Used with Separation Masters	190
4.	The Four Separate Pages Produced from the Example Coding	199
5.	Example of Modifying Part of a GML Profile	202
6.	Example of Assigning the Outside Table Rule to a Separate Master	203
7.	SCRIPT/VS 3800 Font GCM Contents	248
8.	SCRIPT/VS 3800 Font World Trade GCM Contents	251
9.	SCRIPT/VS Fonts: Gothic Text	253
10.	SCRIPT/VS Fonts: Serif Text	254
11.	SCRIPT/VS Fonts: Gothic Highlight	255
12.	SCRIPT/VS Fonts: Serif Highlight	256
13.	SCRIPT/VS Fonts: Gothic Special Purpose	257
14.	SCRIPT/VS Fonts: Serif Special Purpose	258
15.	Sample of Font Library Index Program Listing for the 4250	267
16.	Sample of Font Library Index Program Listing for the 4250	268
17.	Sample of Font Library Index Program Listing for the 3800 Printing Subsystem Model 3	269
18.	Sample of Font Library Index Program Listing for the 3800 Printing Subsystem Model 3	270
19.	Sample of Font Library Index Program Listing for the 3820 Page Printer	271
20.	Sample of Font Library Index Program Listing for the 3820 Page Printer	272
21.	Sample of Font Library Index Program Report for the IBM LaserPrinter 4028	273
22.	Example of a PostScript Code Page	281
23.	Example of Font Entries in the DCFINDEX File	282

Tables

1.	Attributes of a Symbol's Value	39
2.	Required Fonts	71
3.	Characters that Delimit Words for Spelling Verification	86
4.	Code Point Assignments for Accented Characters	90
5.	Dictionary-Maintenance Process Control Card Format	159
6.	Format of an Update Transaction Record	160
7.	Step/DDname Xref	163
8.	Two-Number Code and It's Associated Language	172
9.	Sort Sequences for the Languages Supported by SCRIPT/VS	178
10.	STAIRS/VS Condensed Text Format (CTF) Records	209
11.	Items with Name Changes for DCF Release 4.0	216
12.	Unsupported Control Words	230
13.	Changes to SCRIPT Command Options	231
14.	Changes to SCRIPT/VS Control Words	234
15.	Unsupported TSO/FORMAT Control Words	243
16.	Complete List of Fonts Provided with SCRIPT/VS for the 3800 Printing Subsystem (only for MVS) Fonts Provided with SCRIPT/VS	252
17.	Content of a Font Library Index Structured Field	262
18.	Font Library Index Program Return Codes	266

Summary of Amendments

Release 4.0

Document Composition Facility (DCF), Release 4.0, includes the following:¹

- **Shading:** Allows for shading of boxes, areas, and table cells on Advanced Function Printing (AFP) printers and PostScript devices.
- **MVS and VM Extended Architecture Exploitation:** Allows the DCF program to run and use storage above 16 megabytes.
- **Online Help Command:** Online help for the SCRIPT command is now available for the VM/CMS and MVS/TSO systems.
- **Online Help for Error Messages:** Online GML and SCRIPT error messages are now available for VM/CMS and MVS/TSO systems.
- **Separation Masters:** Allows construction of masters for multiple color printing and multi-part forms.
- **Including Overlays:** Allows you to merge page overlays in a DCF document that is formatted for an AFP printer.
- **Page Segment Enhancements:** DCF allows page segments containing Image Object Content Architecture (IOCA) Image Objects to be included when formatting for some AFP page printers.

The new ABSOLUTE parameter for the .SI [Segment Include] control word allows the user to specify the depth and width of the space to be reserved on the .SI control word instead of using the size of the segment.

- **File Size Limit:** SCRIPT/VS now processes VM input files containing more than 65,535 records.
- **PRINT Option:** Allows for use of the PRINT option on the SCRIPT command in VM. Files sent to AFP page printers² will go directly to SPOOL saving processing time and DASD.
- **IBM LaserPrinter 4028 Support:** Adds support for the IBM LaserPrinter 4028.
- **Font Character Set Size:** Raises the limit on the number of characters allowed in a font character set from 292 to approximately 1000.
- **Language Attribute on the :GDOC tag:** Allows for the selection of the language of the document, which includes translations for literals and messages, and using the correct language for spelling verification, hyphenation, and index sorting sequences in 15 different languages when using the starter set.
- **Hyphenation:** Adds six new languages for spelling verification and hyphenation, and algorithmic hyphenators for all 15 languages.
- **Revision Code Font:** Allows for selecting the fonts used in revision code characters.

¹ Release 4.0 of the Document Composition Facility may be used with Release 3 of the Document Library Facility.

² The PRINT option is still ignored for the IBM 4250 printer and PostScript devices.

- **Revision Character Alignment:** Allows for specifying revision codes on the left or the right or on alternate columns. For a two column format, the revision code character can appear to the left of the first column and to the right of the second column.
- **Table Storage Relief:** The new BREAK parameter for the .TA [Table] control word allows the user to break a table, which places the portion of the table that has been formatted, and releases the storage used for formatting that portion.
- **Page Number Symbol Limit:** Allows for the page numbering limit to increase to 99,999,999.
- **Output Comment:** Allows for use of the .OC [Output Comment] control word with AFP printers. The .OC control word is still ignored when formatting for the 4250 printer.
- **Schedule Tag Set:** Provides a set of tags to allow printing of schedules with start dates, end dates, projected dates, actual dates, slip dates, actions, responsible parties, and schedule changes.
- **Overhead Transparency Tag Set:** Provides a set of tags to allow printing of overhead transparencies with running headers, running footers, and a large, easy to read font.
- **Memo Tag Set:** Provides a set of tags to allow printing of memos in a consistent style.
- **TSO Changes to DCF:** Changes have been made to DCF to improve usability in the TSO environment. The changes can be summarized as follows:
 - Addition of TSO command options
 - Addition of TSO DDNAMES for SCRIPT command files
 - Changes to the Font Library Index Program (FLIP) to allow concatenated font libraries.
- **Module Renaming:** Allows a VM system programmer to change the names used for the DCF disk-resident module, any discontinuous shared segment, and the bootstrap module at installation time.
- **Font Library Index Program (FLIP) Enhancement:** In MVS/TSO, FLIP will search all of the data sets that have been designated as part of a concatenated font library. In VM/CMS, FLIP searches disk extensions for fonts of the specified filetype.

Notices

References in this publication to products or services of IBM do not suggest or imply that IBM will make them available in all countries where IBM does business or that only products or services of IBM may be used. Noninfringing equivalents may be substituted, but the user must verify that such substitutes, unless expressly designated by IBM, work correctly. No license, expressed or implied, to patents or copyrights of IBM is granted by furnishing this document.

Programming Interface

This manual is intended to provide customers with descriptions of text processing, DCF and SCRIPT/VS functions and applications, Generalized Markup Language (GML), installation and customization of its functions and applications, program compatibility, national use considerations, and related publications and products. This manual documents General-Use programming interfaces and associated guidance information provided by DCF.

General-use programming interfaces allow the customer to write programs that obtain the services of DCF.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries:

IBM®	Advanced Function Printing™
PSF™	Print Services Facility™
MVS™	TSO™
VM™	VSE™
BookMaster®	ProcessMaster®
BookManager™	DisplayWrite®
VM/XA™	Virtual Machine/Extended Architecture™
ATMS-III™	IBM 3800 Printing Subsystem Models 3 and 6™
CICS™	IBM 3900 Advanced Function Printer™
AFP™	IBM 3825 Page Printer™
IBM 3827 Page Printer™	IBM 3828 Advanced Function MICR Printer™
IBM 3835 Page Printer™	IBM 3812 Page Printer™
IBM 3816 Page Printer™	IBM LaserPrinter 4028™
IBM 4224-2xx Page Printer™	IBM 4234 Page Printer™
IBM 3820 Page Printer™	IBM LaserPrinter 4019™
IBM 4250 Printer™	

The following terms are trademarks of other companies as follows:

PostScript	Adobe Systems Incorporated
Monotype	Monotype Corporation, plc.
Monotype Times New Roman	Monotype Corporation, plc.
Times Roman	Allied Linotype or its subsidiaries
New Century Schoolbook	American Type Foundry

The fonts provided by IBM are frequently designed by graphic designers in other companies and adapted for IBM printers. The IBM font listed below is a functional equivalent of a font developed by another company:

IBM Font	Functional Equivalent
Sonoran Serif	Monotype Times New Roman

Central Programming Service support and maintenance are available for the following Generalized Markup Language (GML) profiles and macros:

- The GML starter set profile: DSMPROF4
- The GML schedule profile: DSMSPROF
- The GML overhead transparency profile: DSMTPROF
- The GML memo profile: DSMMPROF
- The GML bar-code profile: DSMBPROF
- The GML SCRIPT Mathematical Formula Formatter profile: DSMFPROF
- The GML office document feature profile: DSMOPROF
- The GML office document feature macro-instruction library: DSMOGML
- The GML SCRIPT Mathematical Formula Formatter macro-instruction library: DSMFMAC
- The GML macro-instruction library: DSMGML4

Support and maintenance, however, *are not* available for these profiles and macros if they have been modified in any way. If you modify these items, it is recommended that you also maintain an **unmodified** version of these items for diagnostic purposes.

An authorized program analysis report (APAR) may not be submitted regarding the translation of literals, messages, and phrases in the starter set.

Chapter 1. Introduction to the Text Programmer's Guide

Chapter 1. Introduction to the Text Programmer's Guide	3
Printer References	4
Major Divisions of This Publication	4

Chapter 1. Introduction to the Text Programmer's Guide

This book provides information for the text programmer about the IBM Document Composition Facility (DCF) licensed program and its component text-processing program, SCRIPT/VS. This book should be used in conjunction with the following books:

- *Document Composition Facility: SCRIPT/VS User's Guide*, which shows examples using SCRIPT/VS control words
- *Document Composition Facility: SCRIPT/VS Language Reference*, which defines the SCRIPT command and the SCRIPT/VS control words.

This publication is specifically designed for text programmers whose tasks may include:

- Formatting documents using SCRIPT/VS control words.
- Modifying the DCF Generalized Markup Language (GML) starter set. Refer to the *Document Composition Facility: GML Starter Set Implementation Guide* for more details on modifying the GML starter set.
- Creating GML applications.
- Installing, modifying, and maintaining DCF.

To use this book, you must be familiar with:

- The concepts of text processing and formatting
- The operating environment on which SCRIPT/VS resides
- A text editor supported in the above environment.

This book is not addressed to users of any installation-defined GML applications. These users should refer to their application's documentation. Users of the GML starter set application should refer to the following publications:

- *Document Composition Facility: Introduction to Generalized Markup Language*
- *Document Composition Facility: GML Starter Set User's Guide*
- *Document Composition Facility: GML Starter Set Reference*
- *Document Composition Facility: GML Starter Set Implementation Guide*.

The information in this publication applies equally to MVS, VSE, VM, and CICS³ unless specifically stated otherwise.

³ For information about using CICS, see Appendix F, "DCF Programming Interface for CICS" on page 291.

Printer References

References to the 3800 Printing Subsystem refer to both the 3800 Printing Subsystem Model 1 and to the 3800 Printing Subsystem Models 3 and 6 (in compatibility mode), unless otherwise explicitly stated.

References to the 3800 Printing Subsystem Model 3 also apply to the 3800 Printing Subsystem Model 6 unless otherwise explicitly stated.

References to page printers apply to the following unless otherwise explicitly stated:

- AFP page printers
 - 3800 Printing Subsystem Models 3 and 6
 - 3812 Page Printer
 - 3816 Page Printer
 - 3820 Page Printer
 - 3825 Page Printer
 - 3827 Page Printer
 - 3828 MICR Page Printer
 - 3835 Page Printer
 - 3900 Advanced Function Printer.
 - IBM LaserPrinter 4028
 - IBM 4224-2xx Page Printer
 - IBM 4234 Page Printer

and any other printer compatible at the datastream level.

- 4250 Page Printer
- PostScript devices configured to accept 8-bit ASCII

The IBM 4250 Printer is an electroerosion printer that produces camera-ready copy. The Composed Document Print Facility (CDPF) licensed program is required to print DCF formatted output on the 4250.

Note: Although you can use DCF to format for a 4250 printer, the 4250 Printer and CDPF are no longer available from or supported by IBM.

Although IBM page printers, PostScript devices, and line printers are mentioned throughout this manual, it does not contain detailed information about IBM printers or PostScript devices. For more information read the appropriate introduction and planning guide for the product that interests you.

References to generic logical device types apply to the IBM 3820 Page Printer, the IBM 3812 Pageprinter Model 2, or any advanced-function printers compatible with them at the datastream level.

Major Divisions of This Publication

The chapters of this book are as follows:

- Chapter 1, "Introduction to the Text Programmer's Guide," describes the DCF library and how to use it.
- Chapter 2, "Overview of SCRIPT/VS," describes SCRIPT/VS, how to use it, and what it does.

- Chapter 3, "Defining the Formatting Environment," describes the SCRIPT/VS formatting environment.
- Chapter 4, "Processing Symbols," describes how SCRIPT/VS processes system symbols, passes parameters to input files, and works with system arrays. This chapter also describes how you can assign a value to a particular text variable and offers many useful applications for symbols.
- Chapter 5, "Processing Logical Statements," describes how to alter the order in which input lines are processed. The techniques described include conditional control words, branching, and conditional sections.
- Chapter 6, "Processing Macros," describes the SCRIPT/VS macro-processing capability and how to define a macro, use symbols within a macro, conditionally process parts of the macro, and store macros in a macro library.
- Chapter 7, "Processing GML," describes how to create a GML tag, build an application processing function (APF) associated with the tag, and map the tag to the APF. This section should be read in conjunction with the *Document Composition Facility: GML Starter Set Reference* and the *Document Composition Facility: GML Starter Set Implementation Guide*.
- Chapter 8, "Verifying Spelling," describes how to verify spelling using the SPELLCHK option. This chapter also describes how to build user and addenda dictionaries.
- Chapter 9, "Performance Considerations," contains information on how to improve system performance when formatting documents with SCRIPT/VS.
- Chapter 10, "Tailoring SCRIPT/VS for Your Installation," explains how to modify SCRIPT/VS, how to change SCRIPT/VS system defaults, and how to define a new logical output device or a new font to SCRIPT/VS.
- Chapter 11, "Creating and Maintaining User Dictionaries," describes the dictionary maintenance program that is used to create and modify user dictionaries for checking spelling and hyphenation.
- Chapter 12, "National Use Considerations," describes SCRIPT/VS features related directly to world trade concerns.
- Chapter 13, "Index Sort Sequence," describes the index sort sequences used when creating an index and explains how to change the index sort sequences.
- Chapter 14, "Using Separation Masters," describes how to print separate masters from one output file with the SEPMASTR keyword, SEPMASTR option, and the .SM [Separation Masters] control word.
- Appendix A, "Using SCRIPT/VS with Other IBM Programs," describes SCRIPT/VS compatibility with other programs. It also describes SCRIPT/VS as a postprocessor and as a preprocessor.
- Appendix B, "Compatibility with Earlier Releases of SCRIPT/VS," describes the differences between Release 4 and earlier versions of SCRIPT/VS.
- Appendix C, "IBM 3800 Printing Subsystem Model 1 Fonts Provided with SCRIPT/VS," describes the 3800 Printing Subsystem fonts provided with SCRIPT/VS.
- Appendix D, "The DCF Font Library Index Program," explains how to create font library indexes.

- Appendix E, "PostScript Information," explains PostScript font support and how to add, delete, and customize entries in the PostScript DCFINDEX file.
- Appendix F, "DCF Programming Interface for CICS," explains how to call the SCRIPT/VS formatter as an independent subroutine, using the interface standards described in this appendix.
- Appendix G, "Algorithmic Hyphenation Interface in DCF," describes the interface between SCRIPT/VS Release 3 (and above) and its algorithmic hyphenation routines.
- Appendix H, "Related Publications and Products," includes the following reference publications: publication library guide for DCF, related PostScript publications, and related IBM products.

Chapter 2. Overview of SCRIPT/VS

Chapter 2. Overview of SCRIPT/VS	9
Operating Environments	9
Logical Output Devices and Output Destinations	9
Defaults and Initial Settings	10
SCRIPT/VS Utility Files	10
Communicating with SCRIPT/VS	10
SCRIPT/VS Functions	11
Formatting Functions	11
SCRIPT/VS Programming Facilities	11
Processing Symbols and Macros	11
Processing GML Tags	11
GML Starter Set Application	11
Spelling Verification	12
General Document-Handling Functions	12
Saving Input Lines for Subsequent Processing	12
Specifying the Destination of Output	12
Specifying Color Separation Masters	12
Printing Part of the Output Document	12
Processing Interactively During Formatting	12
Debugging by Tracing Processing Actions	13
Calling the SCRIPT/VS Processor	13
Interactive Environments	13
Batch Environments	13
Using SCRIPT/VS as a Subroutine	13
Using SCRIPT/VS as a Preprocessor	13
SCRIPT/VS as a Processor for RFTDCA	13

Chapter 2. Overview of SCRIPT/VS

SCRIPT/VS can format documents that include SCRIPT/VS control words and Generalized Markup Language (GML) tags as well as text. SCRIPT is the command that you use to invoke the SCRIPT/VS program. SCRIPT/VS can be used with several operating systems in both the interactive foreground and the batch background environments.

Operating Environments

SCRIPT/VS operates in the following interactive environments when the Fore-ground Environment Feature of DCF is installed:

- TSO (Time Sharing Option). SCRIPT/VS runs in the TSO foreground. TSO's interactive editing (ISPF/PDF or EDIT) and related facilities provide the text entry, editing, and data set management that are used with SCRIPT/VS for text processing.
- CMS (Conversational Monitor System). CMS's text editors, file system, and related facilities provide the text entry, editing, and file management facilities that are used with SCRIPT/VS for text processing.
- CICS (Customer Information Control System). An editor, such as Advanced Text Management System-III (ATMS-III), in the CICS environment provides the text entry and editing facilities that are used with SCRIPT/VS for text processing.

With DLF installed, SCRIPT/VS can run in a TSO batch environment under VSE and MVS. DLF is not required to run SCRIPT/VS under MVS. (DLF provides text storage and management for the documents that SCRIPT/VS processes.) In a batch environment, you can also use SCRIPT/VS (under DLF) as a subroutine of a user-written program.

In a batch environment, you can format documents that have been:

- Communicated to the host system from offline word-processing systems or a distributed system.
- Created in a data base/data communication (DB/DC) system.
- Created in an interactive system, such as the VSE/Interactive Computing and Control Facility (VSE/ICCF), and have been submitted with JCL as a job.
- Created in the foreground environments mentioned above (TSO or CICS), but where formatting is more economical in batch than in foreground—for example, a very large document.

Logical Output Devices and Output Destinations

SCRIPT/VS provides flexible composition for printing on terminals, impact printers, or nonimpact printers as an alternative to independent typesetting machines or sending typesetting jobs to a vendor.

When SCRIPT/VS formats a document, it takes into consideration the characteristics of the specific physical output device, called the *logical output device*. This

may be a terminal, a line printer, a nonimpact page printer, or a PostScript device.

A logical device, which is specified with the `DEVICE` option of the `SCRIPT` command, includes a physical device type, a page size, and a number of lines per vertical inch (for line devices). For example, the 3800N8 logical device is a 3800 Printing Subsystem, on a page size of 8.5 by 11 inches.

`SCRIPT/VS` can also be directed to send the output to a specific destination, such as a disk file or the terminal, regardless of the logical device for which it was formatted. You can specify almost any combination of output destination and logical device. For example, when formatting documents that are to be saved for printing at a later date, specify the destination `FILE` and the logical output device of your choice. See the *DCF: SCRIPT/VS Language Reference* for a complete list of available logical devices.

`SCRIPT/VS` can also be used to prepare documents for processing by other programs, such as formatters that support photocomposers, and to prepare data for use as input to the Storage and Information Retrieval System/Virtual Storage (`STAIRS/VS`) licensed program.

Defaults and Initial Settings

`SCRIPT/VS` can format an input file without any control words or GML tags specified. In such cases, the initial settings for page dimensions and formatting controls are used. The initial settings are associated with the logical device specified with the `DEVICE` option of the `SCRIPT` command. The initial settings for each logical device are documented in the *DCF: SCRIPT/VS Language Reference*.

SCRIPT/VS Utility Files

`SCRIPT/VS` creates a number of utility files when it encounters certain control words (such as `.WF` [Write To File]), or command options, or both. You have the option of defining or redefining these files using the `.DD` [Define Data File-id] control word.

Communicating with SCRIPT/VS

You can communicate with `SCRIPT/VS` to determine error conditions and to interactively process your documents.

`SCRIPT/VS` issues program messages accompanied by a severity-level code when certain error conditions are encountered.

By using certain `SCRIPT/VS` control words, you can interact with `SCRIPT/VS` before your input text is in final form and while your document is being formatted. For details, refer to "Communicating with `SCRIPT/VS`" in the *DCF: SCRIPT/VS User's Guide*.

SCRIPT/VS Functions

User-controlled SCRIPT/VS processing includes three general categories of functions: formatting functions, programming facilities, and general document handling.

Formatting Functions

SCRIPT/VS provides you with many text formatting functions, including line composition, page composition, head levels, table-of-contents generation, boxes, keeping text together, footnotes, character translation, indexing, and hyphenation and spelling verification.

SCRIPT/VS Programming Facilities

SCRIPT/VS provides several programming facilities that enable you to specify the SCRIPT/VS formatting environment, process input conditionally, process symbols and macros, and process Generalized Markup Language tags.

You can set the values and parameters of the formatting environment to specify exactly how you want SCRIPT/VS to format each line on an output page.

You can cause SCRIPT/VS to alter input processing. For example, by setting symbol values and comparing those values, you can control whether a block of input text is included in the output document. For details, see Chapter 5, "Processing Logical Statements" on page 43.

Processing Symbols and Macros

You can define symbols and macros for substitution during processing. Symbols are used in many ways: in tests for conditional processing, for cross-references to pages or figure numbers, for entering characters unavailable on the entry keyboard, and as abbreviations for repetitive phrases. You can define macros, which are sets of SCRIPT/VS control words. For example, you might redefine a particular head-level macro to alter the SCRIPT/VS formatting style. For details, see Chapter 6, "Processing Macros" on page 53 in this publication and refer to "Setting Symbols" in the *DCF: SCRIPT/VS User's Guide*.

Symbols and macros are used to support the Generalized Markup Language. The *DCF: GML Starter Set Implementation Guide* describes how symbols and macros are used to create the GML starter set.

Processing GML Tags

SCRIPT/VS recognizes GML tags as a form of text markup and provides extensive facilities for mapping GML tags to APFs, manipulating attributes, and processing symbols. For details, see Chapter 7, "Processing GML" on page 71.

GML Starter Set Application

DCF provides a GML starter set consisting of a profile and a macro library to support a set of tags for general documents. You can either use the starter set as an example of one way to support GML or you can enhance the starter set by adding your tags to suit the needs of your own unique documents. The *DCF: GML Starter Set Implementation Guide* documents how the GML starter set is constructed and illustrates how you can modify it.

Spelling Verification

You can specify whether words are checked for correct spelling. SCRIPT/VS provides dictionaries of many common root words for Dutch, English (Canadian), English (U.K.), English (U.S.), French (Canadian), French, German, Italian, and Spanish. Dictionaries for the following languages are provided, however, they are empty: Danish, Finnish, Icelandic, Norwegian, Portuguese, and Swedish. If you create your own dictionaries for these languages, you can verify spelling against the dictionaries you created.

Algorithms for prefix and suffix variations, provided with all the languages extend the basic root words. SCRIPT/VS determines spelling validity (and hyphenation points) based on these algorithms and the basic root words. You can add words to the addenda or to user-created dictionaries as required for a particular document. For details, see "Spelling Verification" on page 85.

General Document-Handling Functions

SCRIPT/VS provides several document-handling functions, which are described in the sections below.

Saving Input Lines for Subsequent Processing

You can determine whether certain input lines will be written to a file. For details, refer to "Combining SCRIPT/VS Input Files" in the *DCF: SCRIPT/VS User's Guide*.

Specifying the Destination of Output

You can specify the output destination of the formatted document. It can be stored as a file for later use or printed on a variety of devices, including impact and nonimpact printers, and display and typewriter terminals. For details, refer to the description of SCRIPT command options in the *DCF: SCRIPT/VS Language Reference*.

Specifying Color Separation Masters

You can specify that each output page can be separated into multiple pages. This allows you to print separate masters and then mark them indicating that the master should be used in the offset printing process to produce a different color. You can also use this function to produce multi-part forms with simulated text suppression.

Printing Part of the Output Document

You can specify whether every page, a single page, or a range of pages is to be included in the formatted output. For details, refer to the PAGE option in the summary of SCRIPT command options in the *DCF: SCRIPT/VS Language Reference*.

Processing Interactively During Formatting

In an interactive environment (CMS or TSO), you can affect SCRIPT/VS as it processes by entering text or markup at a terminal. In effect, the terminal can be treated as an input file. For example, you can interactively specify the values of symbolic variables specified in the document or enter those portions of text that vary from one processing time to the next. If you are using a typewriter terminal, you can also stop SCRIPT/VS output processing at any point on a line to change typing elements or to enter text. For details, refer to "Interactive SCRIPT/VS Processing" in the *DCF: SCRIPT/VS User's Guide*.

Debugging by Tracing Processing Actions

You can trace all control words and each step of symbol and macro substitution in input lines. In cases where unexpected results are observed, trace information can be an invaluable aid in pinpointing the problem area.

Calling the SCRIPT/VS Processor

You call the SCRIPT/VS processor by issuing the SCRIPT command and specifying the name of the file SCRIPT/VS is to process.

Interactive Environments

In one of the three interactive environments that support SCRIPT/VS, use one of the following formats for the SCRIPT command:

- In CMS: SCRIPT filename (options
- In TSO: SCRIPT dsname options
- In ATMS-III: script docname (options

The SCRIPT command format and options are described in detail in the SCRIPT command options section of the *DCF: SCRIPT/VS Language Reference*.

Batch Environments

For details about calling SCRIPT/VS in a batch environment, refer to the *Document Library Facility Guide*.

Using SCRIPT/VS as a Subroutine

In a batch environment, with the Document Library Facility licensed program, an application program can invoke DLF as a subroutine, which in turn invokes SCRIPT/VS. For details, refer to the *Document Library Facility Guide*.

Using SCRIPT/VS as a Preprocessor

SCRIPT/VS can be used to prepare an input file for use as input to another text program, such as the STAIRS/VS licensed program. For details, see "Producing Input for STAIRS/VS" on page 207.

SCRIPT/VS as a Processor for RFTDCA

When the optional Office Document Feature (ODF) is installed with Release 3.2, DCF can convert an RFTDCA file into an RFT/GML file for formatting by SCRIPT/VS. This feature can also convert an RFT/GML file back into an RFTDCA file. The RFTDCA files can be created on an office system that produces the RFTDCA data stream, such as the DisplayWrite 4 or the DisplayWrite/370.

Chapter 3. Defining the Formatting Environment

Chapter 3. Defining the Formatting Environment	17
Formatting Environment Parameters	17
Running Heading and Footing Environments	17
Keep, Float, Footnote, Table, and Named Area Environments	17
Saving and Restoring the Current Formatting Environment	18
Named Environments	18

Chapter 3. Defining the Formatting Environment

The formatting environment is a set of values and parameters that specify exactly how SCRIPT/VS is to format each line on an output page. The formatting environment consists of three parts:

- The active environment, which contains parameters for formatting text
- The page environment, which contains parameters for defining the entire page
- The translation tables associated with the .TI [Translate Input] and .TR [Translate Character] control words.

Formatting Environment Parameters

When SCRIPT/VS ejects to a new page (or begins the first page), it prepares the output page in the following manner:

1. It saves the active environment values used for body text and initializes the active environment for formatting running headings and running footings.

The active environment is reinitialized before each of these is formatted.

2. Top and bottom page floats are selected from the float queue. If any floats exist and will fit, they are placed on the page.

The output page's running headings and footings, and page floats, are now in place on the output page. All *page control* dimensions are fixed for this page, and any changes to these values will take effect on the next page.

3. SCRIPT/VS restores the active environment for body text that it had saved.

Input lines are processed to produce output lines, which are inserted into the body of the page. When the page is full, or when a page eject occurs, the formatted page is sent to its output destination.

Running Heading and Footing Environments

When running headings and footings are started, SCRIPT/VS:

- Saves the current formatting environment
- Restores the initial formatting environment
- Modifies the environment to reflect changes that have been made with the .DC and .GS control words.

When the running heading or footing ends, the saved formatting environment is restored.

Keep, Float, Footnote, Table, and Named Area Environments

When keeps (other than inline keeps), floats, footnotes, tables, and *named* areas are started, SCRIPT/VS saves a copy of the active environment and then modifies it by:

- Clearing the values of the .OF [Offset] and .UN [Undent] control words
- Restoring the indentation to the basic .IN [Indent] value currently in effect.

In addition, for page floats and footnotes, the column line length is set equal to the line length value.

When the keep, footnote, float, table, or *named* area ends, the saved copy of the active environment is restored.

Saving and Restoring the Current Formatting Environment

The .SA [Save Environment] and .RE [Restore Environment] control words are used to save and restore the current formatting environment. All three parts of the environment are saved and restored by .SA and .RE:

- The active environment
- The page environment
- The .TI and .TR translation tables.

For example, part of an input file that contains a distribution list requires indentation and tab settings to format correctly. However, the main document indentation and tab settings are different. You could use the .SA [Save Environment] and .RE [Restore Environment] control words, as in the following example:

```
.sa
.in
Distribution list for special publications:
.sk
.in 3m
.ti ~ 05
.tp 2i 2.5i
.us Name ~Dept ~Address
.
.
.* End of distribution list
.re
```

to avoid having to restore the main document's values.

Named Environments

The .SA control word saves environments in a stack or by *name*. The .RE [Restore Environment] control word restores the SCRIPT/VS environment to the values that were in effect at the time of the corresponding .SA control word.

To save the current environment by *name*, enter:

```
.sa barnes
```

The current environment is saved as BARNES.

The .SA control word only saves a copy of the values of these SCRIPT/VS variables in the current environment, it does not change any of them.

Because .SA does not change any of the SCRIPT/VS variable settings, all variables should be explicitly set to the appropriate values unless the current settings are known. For example, you can explicitly set indentation to 0 and then restore it to whatever it was previously.

The `.RE` control word restores the SCRIPT/VS formatting environment from a *named* saved environment or from the last-in-first-out stack created by the `.SA` control word. The `.RE` control word restores the SCRIPT/VS variables to values that were in effect at the time of the corresponding `.SA` control word.

To restore the most recently saved *unnamed* environment, enter:

```
.re
```

To restore the *named* environment BARNES, enter:

```
.re barnes
```

If there is no currently active `.SA` control word, the `.RE` control word restores the initial values.

Chapter 4. Processing Symbols

Chapter 4. Processing Symbols	23
Symbols for SCRIPT/VS Control Values	23
&\$ASCI Special Symbol	24
&\$DCF Special Symbol	24
&\$DDUT Special Symbol	24
&\$EGML Special Symbol	24
&\$ENV Special Symbol	25
&\$ENVARRAY Special Symbol	25
&\$GML Special Symbol	25
&\$IXLG Special Symbol	25
&\$LC Special Symbol	26
&\$LST Special Symbol	26
&\$PASS Special Symbol	26
&\$PRT Special Symbol	26
&\$PSNO Special Symbol	27
&\$RET Special Symbol	27
&\$RH and &\$RF Special Symbols	27
&\$SEPM Special Symbol	28
&\$SVLG Special Symbol	28
&\$TAG Special Symbol	28
&\$TAGD Special Symbol	28
&\$VR Special Symbol	28
Passing Parameters to Input Files	29
Setting Symbols with the SCRIPT Command	29
Symbols Set When a File Is Imbedded or Appended	29
Symbols Set When a Macro Is Processed	30
Setting a Symbol to the Current Page Number	31
Symbols for Arrays of Values	31
Controlling the Array Elements	32
Accessing the Index Counter	34
Setting the Index Counter	34
Extended Symbol Processing	35
Defining Text Variables	36
Producing Special Characters	36
Producing a Greek Gamma Character	36
Overriding Delimiter Characters	37
Using Defined Variables to Change Fonts	38
Attributes of a Symbol's Value	39

Chapter 4. Processing Symbols

This chapter describes a number of useful applications for symbols, such as:

- Using system symbols to obtain the current internal formatting parameters.
- Passing parameters from the SCRIPT command to input files, to imbedded or appended files, or to macros.
- Using a symbol to represent the current page number.
- Using symbols for arrays of values.
- Using extended symbol processing.
- Defining text variables.

Symbols for SCRIPT/VS Control Values

SCRIPT/VS allows you to examine the formatting parameter values it uses when processing your input file. You can obtain the current value of a parameter by using the system symbol.

The symbols that represent SCRIPT/VS internal formatting parameters cannot be set by .SE control words in your input file. The name of each of the following reserved symbols begins with &\$ and can be specified using either lowercase or uppercase characters. The system symbols are listed and described in "A Summary of SCRIPT/VS" in the *DCF: SCRIPT/VS Language Reference*.

When a system symbol occurs on an input line, the value substituted is the value of the system symbol that is active before the input line is processed. For example, if the first line in your document is:

The depth of the running heading is: &\$RH.

then the value returned for the system symbol &\$RH will be 0. Prior to the processing of the text line, the page has not yet been started. Therefore, the running heading has not been processed, so the depth of the running heading is zero.

You can specify system symbols to ensure correct results even though some formatting parameters can be changed by control words. For example, the following sequence produces a box the width of the output page:

```
.se indent = &$IN
.se rindent = &$CL - &$IR
.if &indent eq 0 .se indent = 1
.bx &indent &rindent
.in +2m
.ir +2m
The .BX control word begins
a box structure....
.bx off
```

which results in:

The .BX control word begins a box structure using the current margins. The .IN (Indent) and .IR (Indent Right) control words shift the margins to position the text within the box. After the text is processed, the original values are restored.

As another example, you might want to print a page with only a figure caption at the bottom of a single-column page. Perhaps the file is to be formatted within different master files, each of which requires a different page length. You might code the following sequence:

```
.pa
.se lines = &$LC - 1
.sp &lines
Figure x. Sample Output
```

You will find that these special symbols can be especially useful when writing SCRIPT/VS macros or when testing the current environment using the .IF control word family.

&\$ASCII Special Symbol

The &\$ASCII special symbol indicates whether ASCII or EBCDIC was specified on the PSOUT option. The value is 1 if ASCII was specified on the PSOUT option and a PostScript logical device was specified on the DEVICE option. The value is 0 if EBCDIC was specified on the PSOUT option or if a PostScript logical device was not specified on the DEVICE option. ASCII is the default for PostScript devices.

&\$DCF Special Symbol

The &\$DCF special symbol indicates the release and modification level of the module being executed. DCF Release 4.0 has a shipped value of 4.0.0 for Release 4, level 0, and modification level 0.

&\$DDUT Special Symbol

The &\$DDUT special symbol indicates whether the NODDUT or DDUT command option is in effect. The value is either 0, when the NODDUT command option has been given on the command line, or 1, when the DDUT command option has been given.

If neither option has been specified, the value of &\$DDUT is 0, because NODDUT is the default. This special symbol is meaningful only in the CMS and TSO environments.

In ATMS-III and batch, &\$DDUT is always set to 1, indicating that DDUT is the default command option of the pair in these environments.

&\$EGML Special Symbol

The &\$EGML special symbol reflects the value of the current GML end-tag delimiter as set by the .DC [Define Character] GML tag.

&\$ENV Special Symbol

The &\$ENV special symbol indicates the current formatting environment. The range of possible values includes:

AR	Named area
BODY	Body of page
FL	Float
FN	Footnote
FNL	Footnote leader
IBP	In between pages
KP	Keep
RF	Running footing
RH	Running heading
TA	Table

&\$ENVARRAY Special Symbol

The &\$ENVARRAY special symbol provides additional information about the formatting environment. This symbol is an array that shows the different levels of the current formatting environment. The possible values for this symbol are:

AR	Named area
BODY	Body of page
FL	Float
FN	Footnote
FNL	Footnote leader
IBP	In between pages
KP	Keep
RF	Running footing
RH	Running heading
TA	Table

&\$ENVARRAY(0) indicates the number of environments currently active, &\$ENVARRAY(1) indicates the first-level environment, &\$ENVARRAY(2) indicates the second-level environment, and &\$ENVARRAY(3) indicates the third-level environment. &\$ENVARRAY(*) indicates all currently active environments.

For example, if you had some lines of text and entered &\$ENVARRAY(1), the value would be BODY. If a keep was then started, &\$ENVARRAY(2) would have the value KP. If a table was started within the keep, &\$ENVARRAY(3) would have the value TA. &\$ENVARRAY(*) would contain the value BODY, KP, TA. &\$ENVARRAY(0) would have the value 3.

&\$GML Special Symbol

The &\$GML special symbol reflects the value of the current GML tag delimiter as set by the .DC [Define Character] GML tag.

&\$IXLG Special Symbol

The &\$IXLG special symbol indicates the index sort language that will be used to sort all index entries in the document. The value of &\$IXLG can be any of the fifteen supported languages. If the index sort language is set using the .DL control word with the INDEX parameter, the value of &\$IXLG will be that language. If a .DL INDEX control word is not specified, the value of &\$IXLG will be the default language. If a .DL INDEX is then specified, the value of &\$IXLG will

change to the language specified. After the language has been specified using .DL INDEX or indirectly using the .PI [Put Index] control word, the index sort language cannot be changed.

&\$LC Special Symbol

The &\$LC special symbol contains the number of lines left in the column at the time of symbol substitution. The value of &\$LC is most accurate when formatting for line devices. For page devices, be sure and use device units when querying the value of &\$LC (&dv'&\$LC).

This value does not include running headings or footings that have been placed on the page, nor does it include keeps, tables, widows, or partially filled output lines that have not been placed in the column at the time of symbol substitution.

The value of &\$LC at the time of symbol substitution cannot accurately reflect the final position of surrounding text on the page if that text is in a keep, float, or widow; if there is a partially filled output line; or if column balancing is in effect. The value of &\$LC at the time of symbol substitution will accurately reflect the final position on the page of text only at the beginning of a new page, section, or column.

&\$LST Special Symbol

The &\$LST special symbol indicates whether an output line is started. The value is either 1 or 0. It is 1 when text has been formatted (line started), but the line has not yet been completed. It is 0 when a line has been promoted into the current column and more text has not yet been formatted. The value of &\$LST is 0, for example, following a .BR [Break] control word.

&\$PASS Special Symbol

The &\$PASS special symbol indicates the current formatting pass. The value can range from 1–255.

The &\$PASS system symbol can be used with the &\$PSNO system symbol to determine if the current pass is the last pass. For example, you could specify:

```
.if &$PASS eq &$PSNO .se lastpass = yes
.el .se lastpass = no
```

The last pass will be indicated if the current pass (&\$PASS) equals the number of passes specified (&\$PSNO). If the TWOPASS or FPASSES(n) option of the SCRIPT command is not specified, &\$PSNO will be 1, and &\$PASS will also be 1. Therefore, the first formatting pass is the last pass.

&\$PRT Special Symbol

The &\$PRT special symbol indicates whether the current page will be written to the output destination or discarded. Pages are always discarded on the first of two passes when the TWOPASS option of the SCRIPT command is used. Pages can be discarded when the PAGE option of the SCRIPT command is used. The value is either ON or OFF.

&\$PSNO Special Symbol

The &\$PSNO special symbol contains the number of formatting passes requested on the TWOPASS or FPASSES(n) option. The values can range from 1—255. If the &\$TWO system symbol is used with FPASSES(n), and the value specified with FPASSES(n) does not equal 2, &\$TWO is set to zero. Therefore, you should not use &\$TWO to detect the multiple passes when using FPASSES(n) with a value greater than 2.

&\$RET Special Symbol

In CMS or ATMS, the &\$RET special symbol contains the return code from the CMS or CP command that was most recently executed as a result of a .SY [System Command] control word, which is an unsupported control word. Refer to the *Document Composition Facility: Diagnosis Guide and Reference* for information about the .SY control word. You can examine the return code and take conditional action based on its value. For example, the following sequence will imbed a file named OPTDATA only after ensuring that the file exists:

```
.sy state optdata script *  
.if &$RET eq 0 .im optdata
```

In TSO, &\$RET is set to 0 by the .SY [System Command] control word to indicate that the command is stacked for execution after SCRIPT/VS has terminated.

In batch VSE and MVS, &\$RET is set to -3 to indicate that the .SY [System Command] control word is not supported.

&\$RH and &\$RF Special Symbols

The &\$RH and &\$RF special symbols contain the depths of the current running heading or footing, respectively. Initially, the symbols are set to zero. Each time the symbols are used, SCRIPT/VS searches for the current page's running heading or footing depth, and it returns the depth in linespaces of the current font. If &DV' is appended to the front of the symbol name, the depth is returned in device units. The depth may vary when the current page is even or odd if separate even or odd running headings or footings have been defined.

If the depth is queried between pages, before the page is started, or when no running heading or footing is active, the symbol value will be set to zero. For example, if you want to check the depth of the running heading before placing text on the page, you will need to do the following:

```
.pa nostart  
.sp 0  
what is &$RH.
```

The .PA NOSTART control word causes the current page to be ended, but it does not start the next page, and the running heading is not formatted. Because the &\$RH symbol is substituted before the input line is formatted, if the &\$RH symbol is queried in the first line of text, a value of zero is returned. Therefore, the .SP 0 control word is added to cause the page to be started, and to cause the running heading to be formatted.

The &\$RH and &\$RF symbols should not be queried inside the definition of a running heading or footing, that is, between .RH ON and .RH OFF or .RF ON and .RF OFF. This restriction exists because the depths are calculated when the running heading and footing are formatted and placed on the page, not when they are defined. If the symbols are queried inside a definition, a value of zero will be returned.

Symbol values are resolved at the time the symbol is encountered in the input, not when the line is placed on the page. This may cause unexpected results, for example, if the symbol is queried inside a keep or a float. The symbol value will be set to the depth of the running heading or footing on the page where the input line was first encountered, not on the page where the keep or float is eventually placed.

&\$SEPM Special Symbol

The &\$SEPM special symbol indicates what SEPMASTR values are currently active. A series of 17 1s or 0s is returned, representing SEPMASTR 0–16. A 1 indicates that the SEPMASTR suboption was specified. For example, if you specified:

```
SEPMASTR(4 5 8)
```

the &\$SEPM special symbol would return the following:

```
0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0
```

&\$SVLG Special Symbol

The &\$SVLG special symbol indicates the language currently being used for hyphenation and spelling verification. The value of &\$SVLG can be any of the fifteen supported languages. The value of &\$SVLG will be the language specified on the most recent .DL control word. If .DL is not used to set the spelling verification and hyphenation language, the value of &\$SVLG will be the default language.

&\$TAG Special Symbol

The &\$TAG special symbol indicates the name of the last GML tag found. The value is any valid tag name.

&\$TAGD Special Symbol

The &\$TAGD special symbol indicates whether a GML tag or an end-tag was last processed. The value is equal to either &\$GML or &\$EGML.

&\$VR Special Symbol

The &\$VR special symbol indicates whether vertical rules are currently being drawn, because of either the .BX [Box] or the .VR [Vertical Rule] control words. The value is either ON or OFF.

Passing Parameters to Input Files

SCRIPT/VS has three kinds of symbols that are set automatically by parameters passed to a file or macro. These are:

- SCRIPT/VS system symbols, which can be set when the SCRIPT command is issued
- Parameters passed to imbedded files with the .IM [Imbed] and .AP [Append] control words
- Parameters passed to a macro.

Setting Symbols with the SCRIPT Command

Use the SYSVAR option of the SCRIPT command when you want to pass values to the input file from the SCRIPT command line.

The symbols that you can set with the SYSVAR option have names starting with &SYSVAR appended to one alphanumeric character: 0 through 9; uppercase A through Z; and @, #, and \$. For example:

```
script outline ( sysvar ( a atype 2 nogo
```

sets the symbols &SYSVARA to ATYPE and &SYSVAR2 to NOGO. Lowercase letters assigned to an &SYSVAR symbol are translated to uppercase letters. Consequently, when you include the symbols in an input line or as a comparand for an .IF control word line, always use the uppercase symbol name and character-string values.

For example, &SYSVARA can be used to bypass parts of the document, and &SYSVAR2 can be used to terminate processing before completion, as follows:

```
.if &SYSVARA eq ATYPE .go aproc  
.if &SYSVAR2 eq NOGO .qu  
...aproc
```

When you use &SYSVAR symbols, you should put comments at the beginning of your input file so that other users who process the file are aware of each &SYSVAR symbol and the meanings of its values.

For details about the SYSVAR option of the SCRIPT command, refer to the *DCF: SCRIPT/VS Language Reference*.

Symbols Set When a File Is Imbedded or Appended

You can pass parameters to an imbedded or appended file with the .IM [Imbed] and .AP [Append] control words. The symbol &0 is set to the number of parameters passed and the symbols &1 through &14 are set to the parameters following the name of the imbedded file. For example, when the following imbed statement is processed by SCRIPT/VS:

```
.im finance George 125 $21.50 '18-7'
```

the symbols &0 through &4 are automatically set as follows:

<u>Symbol Name</u>	<u>Value Set by SCRIPT/VS</u>
&0	4
&1	George
&2	125
&3	\$21.50
&4	18-7

Symbol &0 contains the number of parameters passed. As many as 14 parameters can be passed when a file is imbedded or appended. These parameters are called *tokens*. Each token can be up to 8 characters long, delimited with blanks. The rules that apply to setting the value of a symbol also apply to specifying a token. Refer to “Combining SCRIPT/VS Input Files” in the *DCF: SCRIPT/VS User’s Guide* for details about imbedding and appending files.

Symbols Set When a Macro Is Processed

You can pass parameters to a macro when your input file calls the macro. The parameters become *local* symbols (that is, symbols that are set only for the called macro, not for other macro calls that occur within the called macro). The format of the macro call might be:

```
.burger fries+shake nosauce 'on a great big poppy-seed bun'
```

When the macro BURGER is processed, local symbols within it are automatically set by SCRIPT/VS:

<u>Symbol Name</u>	<u>Value set by SCRIPT/VS</u>
&*	fries+shake nosauce 'on a great big poppy-seed bun'
&*0	5
&*1	fries
&*2	+
&*3	shake
&*4	nosauce
&*5	on a great big poppy-seed bun

Symbol &* contains the entire untokenized input line. It contains all leading blanks after the blank that delimits the macro name. Symbol &*0 contains the number of symbol values passed. The symbols &*1 through &*n contain the individual tokens passed to the macro. Notice that blanks, arithmetic operators, and parentheses normally delimit tokens, but that a single token can contain these and other special characters if it is enclosed in single quotation marks. Also, macro tokens are not subject to the 8-character limit applied to .IM [Imbed] and .AP [Append] tokens. See Chapter 6, “Processing Macros” on page 53 for details about specifying symbols within macro instructions.

Note: Symbols whose names begin with an asterisk (*) are treated differently from other symbols. Other symbols are globally available to all files and macros, but symbols whose names begin with an asterisk (*) are local to a particular macro at a particular level of nesting. Each time a macro is called, a new set of local symbols is established for it. These symbols are deleted when the macro ends.

Unlike other symbols, local symbols, when undefined, are replaced with the null string during symbol substitution.

Setting a Symbol to the Current Page Number

You can set a symbol to be equal to the value of the current page number when the .SE [Set Symbol] control word is encountered. For example, in the following statement:

```
.se pagenum = &
```

a single ampersand on the right-hand side of the equal sign of a .SE control word is replaced with the character string of the current page number, including its prefix, if any. Elsewhere in your document, you can refer to the page number with its symbol name. To continue the example:

```
For details, see page &pagenum..
```

Whenever the &pagenum symbol occurs in your document, SCRIPT/VS replaces it with whatever the page number was when the .SE [Set Symbol] control word was processed. If the symbol is set before the page is started, the page number will be the same as that of the previous page and not that of the next page. At the start of the document, the page number is 0.

Symbols for Arrays of Values

An array symbol is a special type of symbol that allows you to assign many values to the same symbol name. Each individual element of the array has, in addition to the name, an element number in parentheses. The element number is also called the index or subscript of the element. When you format your document for output, the entire array of values can be referred to by a single symbol name. You can define an array symbol with the .SE [Set Symbol] control word. For example, in the following statement:

```
.se name() = value
```

the parentheses indicate that this is an element of an array and *value* is any expression that can legally appear on a .SE [Set Symbol] control word line. The notation () is a shorthand way to specify the *next* element of the array.

When SCRIPT/VS encounters the array symbol value in the form:

```
&name(*)
```

it replaces &name(*) with the values of all the currently defined array elements, in the order in which they are indexed. A comma and a blank separate the individual elements.

You can specify different array separator characters using the ASEP parameter of the .DC [Define Character] control word. The ASEP parameter allows you to define up to *four* characters that are to be used to separate array elements when an array is substituted in a document using the &name(*) form. All characters to be used in separating array elements must be specified, including blank characters (as hex 40).

The initial array separator characters are the comma (,) and the blank. The characters given with ASEP replace the previous array separator characters.

Suppose you have defined a symbol array containing the names of geographical locations:

```
.se place() = Goldsboro
.se place() = Kunsan
.se place() = Misawa
```

When the symbol `&place(*)` is found, it is replaced with the entire symbol array, and the elements of the array are separated by the array separator characters. For example, the initial setting of the array separator characters is a comma (,) and a blank, so if you enter:

```
At: &place(*)
```

the result of symbol substitution will be:

```
At: Goldsboro, Kunsan, Misawa
```

You can specify up to four characters with the ASEP parameter. You can insert more than four characters, including control words, between array elements by setting the array separator characters to a symbol. For example:

```
.se x = ' and '
.dc asep & x .
At: &place(*)
```

The intermediate result of symbol substitution is:

```
At: Goldsboro&x.Kunsan&x.Misawa
```

The final result of symbol substitution is:

```
At: Goldsboro and Kunsan and Misawa
```

If you want to set the array separator to cause a break, specify:

```
. 'se x = ' ; .br ; '
.dc asep & x .
```

Controlling the Array Elements

Each element in an array has a value associated with it. You can refer to any element of the array with the array symbol name and the element index number in the form:

```
&name(n)
```

where *n* is the positive integer that identifies the position of the element within the array.

An array symbol reference can be used anywhere that a nonarray symbol can be used. If the element *n* exists in the array, its value is substituted just as a normal symbol value would be. If the symbol exists but has no element *n*, a null value is substituted. If the symbol is not defined at all, the symbol is treated as an undefined symbol.

You can specify which array element you want to set by including a number (identifying its location within the array) within the parentheses. For example, the input line:

```
.se list(1) = &
```

sets element number 1 of the array to the current page number. When you list all the elements of the array, this entry will be listed first, even if it is not the first one set. Here is another example:

```
.se name(1) = 1
.se name(47) = 2
.se name(25) = 3
.se name(2) = 4
.se name(3) = 5
```

The expression:

```
&name(*)
```

results in &name(*) being substituted as follows:

```
1, 4, 5, 3, 2
```

In other words, SCRIPT/VS places the array element values in ascending element index order, not in the order in which they were defined. In this example, there are many available but undefined element numbers in between those that are defined. Any undefined elements in an array are ignored when the array values are substituted.

The array element number can be another symbol. For example:

```
.se elem = 1
.se array(&elem) = &
```

No blanks can appear between the symbol name and index. When array symbols are used on the right-hand side of a .SE [Set Symbol] control word expression and symbol substitution is off, symbols used as array subscripts must be simple, not complex, symbols.

An example of a complex symbol is:

```
.se x = ry
.se ry = 4
&&x
```

which resolves first as:

```
&ry
```

and then as:

```
4
```

Accessing the Index Counter

Every array has an element zero, represented by the symbol name:

```
&name(0)
```

Element zero is an index counter that indicates the last element used. It indicates which element should be set next if you did not specify one.

Note: When the TWOPASS option of the SCRIPT command is specified, all array index counters are reset to zero for the second pass, but the individual elements are not reset.

Setting the Index Counter

The expression `&name()` is treated as an index counter as well as a symbolic expression. Each time SCRIPT/VS encounters the expression, it assumes that the next element of the array is to be filled. If you never specify a number within the parentheses of an array symbol, SCRIPT/VS begins numbering with element 1.

It is possible to set the initial value of the array index counter, as follows:

```
.se name(0) = n
```

where *n* is any nonnegative integer. The first occurrence of “.se name(),” with no element specified, would be equivalent to “.se name(*n* + 1)” and the counter would be incremented from there.

In this way, you can start the automatic indexing of an array at element 5, for example, and reserve elements 1 through 4 for explicitly specified definitions.

If you do not set the index counter explicitly, it will be incremented from the index value of the element last set. For example:

```
.se name() = first  
.se name(3) = second  
.se name() = third
```

Element 1 of the array is set to the value *first*, element 2 has a null value, element 3 has the value *second*, and element 4 has the value *third*.

For substitution of arrays, you can cause substitution of all elements in the array (except element zero), or you can cause substitution of just a single element. The notation `&name(5)` causes only element 5 to be substituted. The notation `&name(*)` causes all elements of the array to be substituted, as previously described.

Any symbol is potentially an array symbol. The symbol `&XYZ`, for example, is actually element zero of a possible array. `&XYZ(0)` refers to the same symbolic value as `&XYZ`. If, after using a symbol like `&XYZ`, you set another element with:

```
.se XYZ(5) = 'last letters'
```

be careful about the value previously set in element zero (that is, in symbol `&XYZ`). If the value is not a number, you will get an error message if you ever use the shorthand notation where element zero is supposed to contain the current index.

Extended Symbol Processing

You can place a control word anywhere in an input line as long as it is preceded by the control word separator and a period. You can also invoke a control word or a macro at any point in the input line by setting it as the value of a symbol. This symbol value must also be preceded by a control word separator. When a symbol value begins with the control word separator (;), the rest of the value is treated as though it began a new line. Therefore, a control word that is set as the value of a symbol is processed by SCRIPT/VS as though it were a control word that started in the first character position, even when it occurs in the middle of a text input line. For example, the .BR [Break] control word, defined as the symbol &BR:

```
. 'se BR = ';' .br ;'
```

causes SCRIPT/VS to interpret the symbol &BR as though you had a new input line starting with .br ;. Because the value of the symbol contains a control word separator, the .SE [Set Symbol] control word is entered with the control word modifier (') to inhibit control-word-separator scanning for that input line. Thus, the input line:

```
This is line one.&BR.This is line two.&BR.
```

is formatted as though it were the following four input lines:

```
This is line one.  
.br  
This is line two.  
.br
```

Note: The control word modifier was used here to set up the symbol 'BR' that contained control word separators. The extended symbol processing rule described here takes effect during substitution and not during control-word processing.

Substitution occurs before SCRIPT/VS has classified the line as a control word line or a text line, thus a control word modifier cannot prevent the symbol-substitution processor from recognizing a control word separator.

The input line:

```
.ce Note this; The symbol &BR starts with a semicolon.
```

is formatted as the following four lines:

```
.ce Note this  
The symbol  
.br  
starts with a semicolon.
```

The extended symbol substitution rule only divided the line into three parts. The first part was a control word line (.CE ...) that was later split into two lines by the control word separator rule.

The input line:

```
. 'ce Note this; The symbol &BR starts with a semicolon.
```

is formatted as the following three lines:

```
.!ce Note this; The symbol  
.br  
starts with a semicolon.
```

The control word modifier only suppressed the control word separator rule for the first line after symbol substitution was completed.

For more information about the control word modifier, refer to the SCRIPT/VS control word description section of the *DCF: SCRIPT/VS Language Reference*.

Defining Text Variables

Sometimes you may want to define and assign a value to a particular text variable. For example, you might want to print a special character, such as the Greek gamma character, that is not on your keyboard; or you might want to override a delimiter character so that it is printed as ordinary text; or you might want to change fonts within a GML heading without having to rewrite the APF that defines this heading. The .DV [Define Variable] control word provides a simple way to resolve all of these text-variable concerns.

Producing Special Characters

For many applications, you may need to be able to select any arbitrary character from any arbitrary font, and to do so whenever you want. This can be a problem, however, because the special character you need may not be on your keyboard. Such special characters may include:

- Accented characters for languages other than English
- APL characters
- Greek characters.

Producing a Greek Gamma Character

To produce a Greek gamma character, use the .DV control word. The Greek gamma character, which is hexadecimal 98 in the Pi Serif font, can be defined with .DV for the 3820 Page Printer as follows:

```
.df greek type('pi serif') codepage tlgpi363  
.dv gamma font greek /&X'98  
This is a gamma: &gamma
```

In this example, the symbol &gamma is defined to associate a font (greek) with a text string (hexadecimal 98). The above example results in:

This is a gamma: γ

The symbol &gamma may also be used in the text of a heading as shown in the following example:

```
.h1 the &gamma symbol
```

or as part of a split text control word line such as the following:

```
.sx /&gamma.//greek/
```

See “Using Defined Variables to Change Fonts” on page 38 for more details on using defined variables.

Overriding Delimiter Characters

Another potential problem in SCRIPT/VS is the occasional need to treat one of the special delimiter or escape characters as text. The following characters have special meaning to SCRIPT/VS and need a mechanism to override them if they are to appear as ordinary text characters:

- The *period* (.) delimits a control word when it is in column one of an input line or when it follows a control word separator.
- The *ampersand* (&) delimits a symbol.
- The *GML delimiter* (:) delimits a tag.
- The *page number symbol* (& by default) is replaced by the current page number in running headings and footings.
- The *control word separator* (; by default):
 - Allows several control words to appear on one line
 - Delimits a control word sequence
 - Causes a line to be treated as two logical input lines when it occurs as the first character in a symbol's value.
- The *required blank* (hexadecimal 41 by default) is automatically converted to white space in the output.

In SCRIPT/VS, there are ways to produce any of these characters as ordinary text, but a different mechanism is required to produce each one. In some cases, it is necessary to shut off the normal function of a character in order to use it as text. With text variables, a single mechanism can be used to define symbols for all of these characters, thereby making them available as text characters anywhere in the document. This can be done without shutting off the character's normal function.

In SCRIPT/VS, symbols are commonly defined in a GML profile to allow special delimiters to be used as text characters. For the ampersand, which is the symbol delimiter and the default page number symbol, the symbol &. is defined to be used where a text ampersand is needed. It is done as follows:

```
.se amp &X'08  
.tr 08 &
```

In other words, the symbol &. is replaced by a hexadecimal 08 by substitution, and then all X'08's are translated to ampersands on output.

A simpler method of producing the &. symbol is to use the .DV control word. With text variables, for example, all symbols like &. are defined in this form:

```
.dv amp text /&
```

The **TEXT** form of .DV is used for a character string that has no associated font and that is merely defined as known text. This method has the following benefits:

- No code point need be given up for this symbol
- The translate table is not needed
- The symbol may be used anywhere freely, including in a running heading or footing.

Using Defined Variables to Change Fonts

In SCRIPT/VS there is no mechanism whereby a font can be changed in the middle of certain elements that require all of the text on the same line, such as topic headings or the content of split text control words. You cannot, for example, change fonts in a heading in SCRIPT/VS unless you disregard all of the built-in heading and table of contents functions and rewrite these functions yourself as macros.

A text variable defined with .DV, however, can have a font associated with it, and it can be used freely anywhere, including in a topic heading, table of contents, index entry, or split text. If the text variable requires a change of fonts, this is done automatically as part of the text variable support.

For the 4250, if you want to add a special character from the Pi font to the font of a heading, for example, you could enter:

```
.df pifont type('Pi font Sans Serif' 14) codepage AFTC0363
.dv special font pifont /&x'fc
.h1 &special. Words to the Wise
```

The result of these lines is that the telephone character is printed as the first character in the heading.

Attributes of a Symbol's Value

Attribute	Function
&a'	Converts a numeric character string ¹ to a "base-26" lowercase alphabetic "number."
&A'	Converts a numeric character string ¹ to a "base-26" uppercase alphabetic "number."
&AD'	Returns the depth in line spaces of the material now in the specified <i>named area</i> ² .
&DH'	Converts a space designation ¹ to an equivalent number of horizontal device units.
&DV'	Converts a space designation ¹ to an equivalent number of vertical device units.
&E'	Verifies the existence of a symbol; the value is 1 if the symbol has been set; 0 if not.
&L'	Returns the number of characters in a character string. ¹
&r'	Converts a numeric character string ¹ to a lowercase roman numeral.
&R'	Converts a numeric character string ¹ to an uppercase roman numeral.
&SD'	Returns the depth in line spaces of a named segment to be included with the .SI control word ² .
&SH'	Returns the horizontal measurement units of a named segment to be included with the .SI control word.
&SV'	Returns the vertical measurement units of a named segment to be included with the .SI control word.
&SW'	Returns the width in figure spaces in the default font of a named segment to be included with the .SI control word ² .
&T'	Yields the type of the current value of a symbol. The type is either "N" for numeric or "C" for character.
&U'	Converts a lowercase character string to uppercase.
&V'	Yields the current value of a symbol.
&W'	Yields the measured length of a character string ¹ in figure spaces of the default font ² .
&X'	Converts a hexadecimal notation to a character string ¹ .
¹ The character string or space designation may be the value of a symbol. ² If you want the length of a character string in device units, use &DV' or &DH' in combination with this attribute.	

Table 1. Attributes of a Symbol's Value

Chapter 5. Processing Logical Statements

Chapter 5. Processing Logical Statements	43
The .IF Control Word Family	43
Alternative Processing	44
Bypassing Part of an Input File	45
The SYSPAGE and SYSOUT Comparands	46
Special Techniques for Conditional Processing	46
Conditional Sections	47
Logical Processing With Symbols	49

Chapter 5. Processing Logical Statements

SCRIPT/VS provides several methods for processing input logically or conditionally. You can write input files and macros that are capable of making simple decisions and taking action based on the result. With logical processing techniques, you can do the following:

- Select the alternative input lines to be processed in a particular run.
- Construct loops that process the same material several times to provide several copies of the formatted output. (Each copy can, of course, contain different specific information.)
- Write macros that cause different formatting based on the logical output device or other variables.
- Provide processing based on the *content* of an input line.

These capabilities use basic logical processing techniques in conjunction with other techniques that are not described here. Chapter 6, "Processing Macros" on page 53 contains information about the mechanics of writing macros, and Chapter 4, "Processing Symbols" on page 23 describes symbol substitution. Individual control words are described in the SCRIPT/VS control word description section of the *DCF: SCRIPT/VS Language Reference*.

There are three basic logical processing techniques:

- The .IF control word family
- Conditional sections
- Conditional processing with symbols.

The .IF Control Word Family

SCRIPT/VS allows you to test a symbol value to determine whether to process an input line or to ignore it. To make this conditional test, you can use the .IF [If] control word alone or in conjunction with the .AN [And], .OR [Or], .TH [Then], and .EL [Else] control words. Using the .IF [If] control word alone is the simplest way of specifying a conditional statement. This control word is specified in the form:

```
.if comparand1 test comparand2 target-line
```

Each comparand can be up to 255 characters long⁴. The shorter comparand will be padded with blanks to match the length of the longer comparand.

⁴ The entire input line, after substitution, cannot be longer than 255 characters. When comparing symbols that can potentially have long values or contain blanks, we recommend that the .IF control word be performed with substitution off, as described in "Special Techniques for Conditional Processing" on page 46.

The conditions you can test for and the codes you can use are:

Code	Meaning
= or eq	equal to
!=, <>, or ne	not equal to
> or gt	greater than
< or lt	less than
>= or ge	greater than or equal to
<= or le	less than or equal to

The target-line part of the .IF [If] control word can be any valid SCRIPT/VS input line: a control word, a symbol, a macro, or text. The first nonblank character after comparand2 is treated as the first character of the input line. If the condition is true, the input line is processed by SCRIPT/VS. Otherwise, it is ignored.

Alternative Processing

There may be times when, depending on the results of a comparison, alternative processing can occur. You can use multiple .IF [If] control words to handle this situation, or you can use the .TH [Then] or .EL [Else] control words in conjunction with an .IF control word. For example:

```
.if &street eq Broadway .se branch = Commercial
.if &street ne Broadway .se branch = Warehouse
```

causes the same result as:

```
.if &street eq Broadway
.th .se branch = Commercial
.el .se branch = Warehouse
```

Both of these examples result in the symbol &branch being set to the value *Commercial* if the comparison is equal and to the value *Warehouse* if it is not.

The .TH [Then] and the .EL [Else] control words cause their targets to be executed or ignored based on the result of the most recently executed comparison in the current file or macro. Therefore, a series of conditionally executed lines can follow a single comparison. For example:

```
.if &job eq chimney-sweep
.th .sp 2
.th .notes height of roof
.el .us salary
.el .in 5
.el .im salary &job
.th .sp 2
.el .sp 4
```

causes all of the .TH and .EL control words that follow the .IF control word to be executed or ignored based on the result of its comparison. Other .IF control words that can be contained in the .NOTES macro or the SALARY file do not affect this series of control words, because the result of the most recent comparison is preserved across macro calls and imbedded files.

There may also be times when you want to test for multiple conditions. You can accomplish this by using the .AN [And] and .OR [Or] control words in conjunction with the .IF control word. For example, you might have a situation where two conditions have to be true before a certain type of processing can occur. In this situation, specify:

```
.if x&city = xFayetteville .an x&state = xArkansas .se zip = 72701
```

which causes the symbol &zip to be set to 72701 if both conditions are true. The character 'x' is added prior to each comparand in the .IF or .OR control word statements to handle the possibility of &city or &state having a null value. If this technique is not used and the symbols have a null value, the control word line would be substituted. The symbols would look like the following when the control word line is processed:

```
.if eq Cleveland .or Arkansas .se zip = 72701
```

When the control word line is processed, the string "eq" is interpreted as the first comparand, and the string "Cleveland" is interpreted as the test between the comparands. An error will occur, because "Cleveland" is not one of the valid tests.

Similarly, you can have a situation where only one of multiple conditions must be true for one type of processing to be done. In this case, you might specify:

```
.if x&city eq xCleveland .or x&city eq xWarrensburg  
.th .carpool &city  
.el .se city =
```

The macro .CARPOOL will be invoked if the value of the variable &city is either *Cleveland* or *Warrensburg*; if it is neither, the variable &city will be reset to null.

Bypassing Part of an Input File

When you want to bypass a part of your current input file, you can use the .GO [Goto] and ... [Set Label] control words. For example:

```
.if &type = 1 .go bypass  
.  
.  
.  
...bypass
```

In the above example, if the symbol &type has a value of 1, all the control words and text between the .IF and the ... [Set Label] control words (which sets the label *bypass*) are skipped.

Conditional processing with the .IF [If] control word can be especially convenient when one file is imbedded in several different master files. You can provide for slight differences among the files by setting the same symbol to a different value in each master file and using that symbol to determine how processing is to be done in the imbedded file.

The .GO function, on the other hand, can be relatively inefficient. You should restrict its use to situations where it best achieves the required results. When the label follows the .GO in your input file, processing is most efficient if that label is near the .GO; when the label precedes the .GO in your input file, processing is most efficient if that label is near the beginning of the file.

Label processing in macros is much more efficient than in files. However, it is most efficient to branch to a label that is near the top of a macro, because the search for labels always begins at the top.

The SYSPAGE and SYSOUT Comparands

You can use the SYSPAGE and SYSOUT comparands with the .IF [If] control word family. They are keywords, not symbols; therefore, they should not be pre-faced with an ampersand (&).

- SYSPAGE tests whether the page currently being formatted is an even- or odd-numbered page (EVEN or ODD). You can use SYSPAGE to place text on an output page, based on whether the output page is even-numbered or odd-numbered:

```
.if SYSPAGE = EVEN .sx /Evenpage Top Line///  
.if SYSPAGE = ODD .sx ///Oddpage Top Line/
```

- SYSOUT tests whether the destination of the output is the line printer (PRINT), page printer (PAGE), PostScript device (PS), or terminal (TERM). This keyword is provided for compatibility with SCRIPT/370. The SCRIPT/VS system symbols &\$LDEV and &\$PDEV provide a better way to test which of the many logical and physical output devices possible with SCRIPT/VS is currently in use.

Special Techniques for Conditional Processing

There are several techniques you should be aware of when using the .IF [If] family of control words.

- Comparing Null-Value Symbols

When you specify the name of a symbol value that might be null, you should prefix the symbol name with a character-prefix to avoid a possible syntax error. For example, the input line:

```
.se a = ''  
.if &a = ON .go next
```

results in a SCRIPT/VS error because the symbol &a was set to a null value. The conditional statement resolves to:

```
.if = ON .go next
```

The = character is treated as the first comparand, and ON is not a valid comparison. However, the input line:

```
.if /&a = /ON .go next
```

resolves to:

```
.if / = /ON .go next
```

When the symbol &a has the value ON, it resolves to:

```
.if /ON = /ON .go next
```

That is, the prefix / is concatenated with the value of &a to result in /ON, which satisfies the test. When the symbol &a is null, /&a results in / and the test fails, but no error results.

- Comparing Symbols Containing Special Characters

The .IF [If] control word family, like the .SE [Set Symbol] control word, is capable of resolving symbols in its comparands even if symbol substitution is off. This capability is essential when comparing symbols whose values might contain special characters, such as blanks and control word separators, or whose values might be long. For example, with symbol substitution on, the input line:

```
.if &needle eq &haystack .th .im lost
```

might result in:

```
.if Rachel's MG eq Parking Lot .th .im lost
```

after symbol substitution has occurred. This would result in an error because Rachel's would be interpreted as the first comparand and MG would be interpreted as an invalid comparator. With substitution off, the symbols &needle and &haystack will be recognized as the comparands, and symbol substitution will be performed on the two comparands separately before they are compared.

- Comparing Potentially Long Comparands

After substitution, an input line cannot be longer than 255 characters. If your input line might exceed 255 characters after substitution has been performed, the .IF control word should be processed with substitution off.

Conditional Sections

When a document might be read by several different audiences, you can customize it for each. To do this, you identify those sections of the input file that are to be processed conditionally.

SCRIPT/VS processes a conditional section, or ignores it, based on the setting of a .CS [Conditional Section] control word. Each conditional section number, from 1 to 9, can be used many times in a document. You can associate each type of information to be processed conditionally with its own conditional section number. For example:

Conditional Section Number	Conditional Section Applies To
1	Only Class A Widgets
2	Only Class B Widgets
3	Only Class C Widgets
4	Either Class B or Class C (Not Class A)
5	Either Class A or Class C (Not Class B)
6	Either Class A or Class B (Not Class C)

At the beginning of the document, specify that SCRIPT/VS is to bypass all conditional sections with the IGNORE parameter of the .CS [Conditional Section] control word. The SCRIPT/VS default is to process all conditional sections not specifically bypassed.

```
.cs 1 ignore
.cs 2 ignore
.
.
.
.cs 6 ignore
```

Before you issue the SCRIPT command to process the document, change some of the .CS [Conditional Section] IGNORE control words to .CS [Conditional Section] INCLUDE control words, to process each desired conditional section. For example, to print all material appropriate for readers interested in Class B Widgets, specify:

```
.cs 2 include
.cs 4 include
.cs 6 include
```

In the body of your input file, you identify each conditional section by preceding it and following it with the .CS [Conditional Section] control words, using the ON and OFF parameters. For example:

```
.cs 2 on
This material applies only
to Class B Widgets.
It does not apply to either
of the other types.
.cs 2 off
```

Because you can specify only one conditional section number with the .CS [Conditional Section] control word, you must use a separate number to identify sections that apply to either one of two (but not the third) types of device.

Because the .CS [Conditional Section] control word does not cause a break, you can process small units of text conditionally. For example, the input lines:

```
.cs 1 ignore
.cs 2 ignore
.cs 3 include
This book is written specifically
for the operator of a
.cs 1 on
Class A
.cs 1 off
.cs 2 on
Class B
.cs 2 off
.cs 3 on
Class C
.cs 3 off
Widget.
```

are printed as:

```
This    book    is    written
specifically for the operator of
a Class C Widget.
```

The input lines (GML tags, control words, and text) between the .CS ON and the .CS OFF control words are included unless explicitly bypassed as a result of a preceding .CS IGNORE control word. Such a bypass is not a total one: macros and GML tags are resolved.

Logical Processing With Symbols

With set symbols, you can do logical processing in several ways. The simplest of these is to have a symbol that resolves to one control word or another, depending on the specific, applicable conditions. For example, the symbol *xim* could be set to either .CM or .IM to cause the input line:

```
&xim filename
```

to be treated as an .IM [Imbed] control word or as a .CM [Comment] control word. If your file has several places at which another file should be imbedded conditionally, the symbol *xim* could be defined once to control all occurrences of the symbolic control word.

Another technique uses the existence attribute (&E') of a symbol to generate a macro name according to whether or not a symbol exists. See Chapter 4, "Processing Symbols" on page 23 for details on symbol attributes. The existence attribute causes a string to be substituted with 0 if a symbol does not exist and with 1 if it does. You could write a macro called X0 to provide the appropriate processing when a given symbol does not exist, and another called X1 for when it does exist. Now, the expression:

```
.X&E'&name
```

will resolve to .X0 if the symbol *&name* does not exist and .X1 if it does.

You can also use the symbol length attribute (&L') to perform logical processing. The length attribute and the following string or symbol are replaced with the length of the string or symbol during substitution. See Chapter 4, "Processing Symbols" on page 23 for details. If a symbol called *&num* contains a number that is from one to five digits long, you can develop a 5-digit number by adding the correct number of leading zeros to *&num*. First, you need to define symbols that contain the number of zeros needed for each possible length the number might be:

```
.se 5z =  
.se 4z = 0  
.se 3z = 00  
.se 2z = 000  
.se 1z = 0000
```

If the number is five digits long, zeroes need not be added. If it is four digits long, you need one zero, and so on. Now, the expression:

```
&&L'&num.z.&num
```

concatenates the correct number of zeros to the number to form a 5-digit number. One part of the expression, &L'&num, is resolved to the number 1, 2, 3, 4, or 5, whatever the length of the number in the symbol *&num* happens to be. If it is 3, the expression becomes &3z.&num. The symbol &3z is now replaced with 2 zeros, the correct number of zeros for a 3-digit number, and is concatenated with the number itself when *&num* is substituted.

Chapter 6. Processing Macros

Chapter 6. Processing Macros	53
When to Use Macros	53
How to Define a Macro	54
How Values Are Substituted for Symbols within a Macro Definition	56
Conditional Macro Processing	57
Macro-Naming Conventions	57
Local Symbols for Macros	57
Processing Local Variables	59
Terminating a Macro	59
Redefining SCRIPT/VS Control Words	60
Avoiding an Endless Loop	61
Using Symbols and Macros as Associative Memory	61
Redefining SCRIPT/VS Formatting Conventions	63
Processing Input Lines That Begin with a Blank or a Tab	63
Specifying a Macro Library	64
LIB: Specify Symbol and Macro Libraries	64
Creating SCRIPT/VS Macro Libraries	66
In a CMS Environment	66
In a TSO Environment	66
In an ATMS-III Environment	67
In a DLF Environment	68

Chapter 6. Processing Macros

SCRIPT/VS allows you to define your own processing controls, called macro instructions. A macro instruction can consist of SCRIPT/VS control words, GML markup, symbols, text lines, and other macros.

You can define macros for GML processing to provide additional formatting controls or to modify the action taken by a SCRIPT/VS control word.

To process macros, you must explicitly specify .MS [Macro Substitution] ON in your document before SCRIPT/VS encounters any of the macros. If SCRIPT/VS encounters a macro when macro substitution is off, the first two characters of the macro are treated as a control word.

When to Use Macros

Many macro-like functions can be performed by symbols that are defined as control word strings. Sometimes, though, you may need to define a macro to perform a function that symbol processing alone cannot provide. For example, the control word sequence:

```
.se x = &x + 1;.se y = &x
```

is intended to increment the symbols x and y. But because SCRIPT/VS performs symbol substitution before control word execution, &y is set equal to the current value of &x, and only &x is incremented.

You can perform this sequence correctly by defining a macro. For example:

```
.su off  
.dm increment /.se x = &x + 1 /.se y = &x  
.su on
```

After SCRIPT/VS processes the macro:

```
.increment
```

&x and &y have equal values, because the two .SE [Set Symbol] control words are processed sequentially.

Macros also allow you to redefine the meaning of SCRIPT/VS control words. For example, you can use the macro facility to define new head levels. Although seven head levels, .H0—.H6 [Head Level 0—6], are provided with SCRIPT/VS, you might want to define additional head levels.

How to Define a Macro

Use the .DM [Define Macro] control word to define macros. Because SCRIPT/VS processes macros as control words, an undefined SCRIPT/VS macro may be treated as an invalid control word.

When you define a SCRIPT/VS macro, you must name the macro and specify the input lines to be processed whenever the macro is called. You can write the following paragraph macro:

```
.su off
.dm para /.sk /.in 3 for 2 /&*
.su on
```

The macro definition elements (usually control words) are separated by delimiters. The delimiter is the first nonblank character that follows the blank after the macro name. It can be any character that does not appear in the line itself.

The symbol &* represents the entire macro argument (that is, the line passed to the macro for processing). For example, when the input line:

```
.para On second thought,
```

is processed, &* has a value of *On second thought*.

The form of the .DM [Define Macro] control word shown above is restricted to one input line. The input line is broken at delimiter characters into separate macro lines.

The simplest way of defining a macro within a document is this:

```
.dm echo on
.ty ===
.ty &*
.ty ===
.dm off
```

The inline form (ON/OFF) of the .DM [Define Macro] control word allows you to define macro lines on separate input lines. For example: you could define the .PARA macro as follows:

```
.dm para on
.sk
.in 3 for 2
&*
.dm off
```

All of the input lines between the .DM PARA ON and the .DM OFF will be put into the macro definition. Substitution and input translation will *not* be performed on these lines until the macro is invoked. The .DM OFF control word line must begin in column one of the input record and cannot contain the control word modifier. The inline form of the .DM control word requires that you completely define the macro each time.

This type of macro definition includes an implied .SU OFF and .SU ON and .GS TAG OFF and .GS TAG ON. Symbols and GML tags in the macro definition are not resolved but are saved as part of the macro and then substituted whenever the macro is executed. This eliminates the need to surround a macro with .SU OFF and .SU ON and with .GS TAG OFF and .GS TAG ON when using this form of the .DM control word.

Another method of defining macros involves using the subscripted form of the .DM control word. For example:

```
.su off
.dm para(5) /.sk
.dm para(10) /.in 3 for 2
.dm para(15) /&*
.su on
```

The macro line number in parentheses is also called the subscript. If the number is omitted from the parentheses, SCRIPT/VS uses an increment of 10, starting at 10. Macro line numbers, if included, do not have to be defined in any particular order. However, when the macro is used, it is executed in subscript order, which is not necessarily the sequence in which the macro lines were entered.

Each line can be defined separately, and each line can be given an explicit line number. Each macro line can contain several control words, separated by control word separators. If so, the control word separator scan must be prevented. This can be accomplished by using the control word modifier on the .DM control word.

```
.su off
.'dm echo(1) /.ty ===; .ty &*
.dm echo(3) /.ty ===
.su on
```

If line numbers are not given explicitly, they will be automatically generated, using an increment of 10:

```
.su off
.dm echo() /.ty ===
.dm echo() /.ty &*
.dm echo() /.ty ===
.su on
```

Whenever the single-line or subscripted forms are used, remember to turn symbol substitution off during the definition, so that symbol names will be saved as part of the macro. You also need to turn GML processing off with .GS TAG OFF if there are GML tags in the macro.

The subscripted form of the .DM control word can be used to modify individual lines of a macro without having to respecify the entire macro definition. For example, to increase the indention caused by the previously defined .PARA macro, you can issue:

```
.dm para(10) /.in 5 for 1
```

or you can cause the .PARA macro to start an inline keep by specifying:

```
.dm para(12) /.kp 6
```

Be careful when you mix the forms of the .DM control word. You can use the single-line form and the subscripted form within an inline macro definition, but you cannot use an inline macro definition within another inline macro definition.

How Values Are Substituted for Symbols within a Macro Definition

Macro definitions almost always contain symbols, and these symbols are generally meant to be substituted anew each time the macro is executed. You can ensure that symbol substitution is turned off when you define a macro within a document and that it is turned on when you execute it, by using the .DM macro-name ON form to define macros. This form explicitly disables symbol substitution for the definition:

```
.dm count on
.se x = &x + 1
.ty &x
.dm off
```

The symbol &x will be saved as part of the macro definition and will be substituted whenever the macro is executed. Each time the macro is executed the value of x will be different.

If you use the subscripted or the single-line form of the .DM [Define Macro] control word and symbol substitution is on, the .DM [Define Macro] control word line is scanned for symbol names. If you define a macro that contains a symbol, you usually want the symbol value substituted for the symbol name when the macro is encountered as an input line, rather than when the macro is defined. Therefore, turn off symbol substitution (using the .SU OFF control word) before you define the macro, to allow the symbol (rather than its value when the macro is defined) to be part of the macro definition. In the following example:

```
.su off
.dm of /.sk/.in &off after 1/
.su on
```

&off is a symbol that might have a value when SCRIPT/VS processes the .DM [Define Macro] control word. If substitution is ON, the symbol value becomes part of the macro definition instead of the symbol &off. The macro .OF would then result in a hanging indention of that amount, rather than of the value of &off when SCRIPT/VS encounters the macro .OF.

When you use the inline form of the .DM control word, you do not have to be concerned with when symbol substitution will be performed on the macro lines. It will not be done when the lines are read into the macro definition, only when the macro is actually executed.

Conditional Macro Processing

Macros can be defined to conditionally format a document using the `.IF [If]` control word family. For example, you can have a series of input files that contain information for several people, none of whom require all of the information. You can define a macro that will execute certain control words only if the document is being formatted for specific individuals:

```
.dm canbe on
.if &who eq Geoff .or &who eq Dot
.th &*
.dm off
```

When you specify:

```
.canbe .im pg$sym
```

the file `PG$SYM` will be imbedded only if the document is being formatted for either Geoff or Dot. The symbol `&who` has presumably been set elsewhere.

You can also use conditional processing to highlight lines of text differently, depending upon the device for which the document is being formatted. The following macro will cause a line of text to be put in uppercase, if the document is being formatted for a 3270 terminal. For all other devices, the line will be underscored.

```
.dm hilite on
.sk
.if &$PDEV eq 3270
.th .up &*
.el .us &*
.sk
.dm off
```

Macro-Naming Conventions

A macro name can be up to 10 nonblank characters long, cannot have imbedded blanks or special characters, and is not case sensitive. That is, the macro name `ABC` is the same as the macro name `abc`. The name can be the same as the two-letter name of a control word, in which case its definition supersedes the function of the control word. When you enter a macro name as part of your input file (after you have defined it), enter it as though it were a control word, with a period in column 1.

Local Symbols for Macros

Within macros, symbols can be defined with an asterisk (*) as the first character of the symbol name. Such symbols are *local* to the macro in which they are defined. They are recognized only within that macro and, unlike ordinary symbols, if they are undefined, they have a null value. You use a different set of local symbols for each macro, and for each occurrence of a macro call.

For symbol substitution within a macro, the following rules apply:

- All global symbols are considered text character strings if undefined as symbols.
- All local symbols are considered null if not defined.

When SCRIPT/VS processes a macro, it assigns values to certain designated local symbols, based on the macro's input line. The local symbols are named `&*0`, `&*1`, `&*2`, and so on. Values are assigned to a new set of local symbols each time a macro is called.

The symbol `&*` contains the entire character string on the macro's input line (except for the macro name). The symbol `&*0` represents the number of words or tokens that make up the character string. The symbol `&*1` contains the first token, the symbol `&*2` contains the second token, and so on. For example, when SCRIPT/VS encounters the following input line:

```
.process fileb 10 filea no
```

it sets the following values for the macro's local symbol values (`&*`, and `&*1` through `&*n` are called *tokens*):

<u>Symbol</u>	<u>Value</u>
<code>&*</code>	fileb 10 filea no
<code>&*0</code>	4
<code>&*1</code>	fileb
<code>&*2</code>	10
<code>&*3</code>	filea
<code>&*4</code>	no
<code>&*5-<i>&*n</i></code>	(null value)

When you want to assign a null value to a macro symbol without also assigning null values to all subsequent tokens on the input line, use the percent sign (%) to represent the null-value token. For example, the macro input line:

```
.insert filea 10 % fileb 15
```

results in the symbols being set as:

<u>Symbol</u>	<u>Value</u>
<code>&*</code>	filea 10 % fileb 15
<code>&*0</code>	4
<code>&*1</code>	filea
<code>&*2</code>	10
<code>&*3</code>	(null value)
<code>&*4</code>	fileb
<code>&*5</code>	15
<code>&*6-<i>&*n</i></code>	(null value)

You can set any symbol with a name that begins with the character `*`. A symbol so named is considered a local symbol for the macro whose definition includes it. Such symbols are known only to the macro that defines them. The symbol values are saved when the macro calls another macro and are restored when the called macro returns to the calling macro. A different set of local symbols is set each time a macro is called, plus another set for when no macro is the current source.

Note: Undefined local symbols are replaced with null values only when the current input source is a macro.

Processing Local Variables

Macros can contain conditional and iterative processing and can use local variables. You can use the `.GO` [Goto] control word to branch to another portion of a macro on certain conditions. For example, to process each token specified with a macro invocation separately you could enter:

```
.dm macro on
.se *i = 1
...loop
.process &*&i
.se *i = &*i + 1
.if &*i le &*0 .go loop
.dm off
```

Terminating a Macro

Ordinarily, processing of a macro ends after the last line of the macro has been processed; control returns to the file or macro that invoked the macro.

The `.ME` [Macro Exit] control word can be used to end processing of a macro prematurely:

```
.dm score on
.sk 1
.if &place eq inline .me
.hr left to right
.sk 1
.dm off
```

If the value of the symbol `&place` is *inline*, the `.ME` control word causes control to return immediately to the macro's caller, without processing the remainder of the macro.

If the remainder of an input line containing a `.ME` [Macro Exit] control word is not null, it is saved until after the macro is closed and then is processed as if it had been part of the calling macro or file. This allows a macro to set its caller's local symbols. For example:

```
.dm macro on
.
.
.
.me .se *rc = 4
.dm off
```

Here the `.ME` control word function of prematurely ending the macro is superfluous, because it is the last line of the macro. The remainder of the line, however, is saved and executed as if it had been part of the macro's caller, and results in the setting of a macro local symbol.

The `.ME` [Macro Exit] control word also allows you to create a *computed GOTO* facility:

```
.dm case on
.se *i = &*1 + 1
.if &*i gt &*0 .mg ||CASE index error.|
.el .me .go &*&i
.dm off
```

The CASE macro can be invoked with an index number (0, 1, 2, or 3) and a list of labels:

```
.case &function open read write close
```

The CASE macro uses the index to select one of the labels and return a .GO [Goto] control word for that label to its caller.

Redefining SCRIPT/VS Control Words

You can define a macro with the same name as a control word to effectively redefine it, to revise it, or to supplement its function. The definition you code with the .DM [Define Macro] control word is used instead of the SCRIPT/VS-defined function. If you redefine a control word as a macro, the new definition is effective whenever the control word is encountered as long as macro substitution is on (.MS ON), or whenever the macro is called using the .EM [Execute Macro] control word.

When macro substitution is on, you can still specify that a SCRIPT/VS control word function is to be executed, even when a macro of the same name is defined, by using the .EC [Execute Control] control word or the control word modifier. For example, the input line:

```
.dm sk on
.ec .sk &*
.il 5
.dm off
```

redefines the .SK [Skip] control word, to skip lines and indent the first output line after the line space.

When you want the .SK [Skip] control word to be executed but do not want to turn off macro substitution, issue:

```
.ec .sk 4      or      .'sk 4
```

to skip four lines without indenting the next output line (that is, to execute the control word rather than the macro).

When macro substitution is off (.MS OFF) and you want to execute a macro (whether or not the macro name is the same as a SCRIPT/VS control word), use the .EM [Execute Macro] control word. For example:

```
.em .sk 3
```

results in three line spaces, with the next output line indented five spaces.

Note: When you redefine a SCRIPT/VS control word with a macro of the same name:

- Be sure to define all the functions, implicit as well as explicit, that you want. The macro definition *does not* modify the control word function; it is used, as a macro, *instead of* the control word function.
- To make the macro definition effective, either turn macro substitution on (.MS ON) or use the .EM [Execute Macro] control word to execute the macro definition.
- When the macro definition includes the SCRIPT/VS control word of the same name, use the .EC [Execute Control] control word to specify the control word. An example of this technique is in the following section, “Avoiding an Endless Loop.”

Avoiding an Endless Loop

When you define a macro to replace the function of a SCRIPT/VS control word, you might have to turn macro substitution off to avoid an endless loop. For example, if you want to redefine the .SK [Skip] control word to skip lines and indent to the first output line:

```
.dm sk on
.ms off
.sk &*
.ms on
.il 5
.dm off
```

Because we turned macro substitution off with the .MS OFF control word, the third line of the macro invokes the .SK control word, rather than reinvoking the .SK macro.

Sometimes turning off macro substitution is not an adequate solution to the problem of an endless loop. For example, you can cause the .IM [Imbed] control word to type the name of the imbedded file whenever it is imbedded by defining an .IM macro:

```
.dm im on
.ty Imbedding &* . . .
.ms off
.im &*
.ms on
.dm off
```

Macro substitution is turned off to prevent an endless loop from occurring. However, when macro substitution is turned off, substitution is prevented for any macro that might be part of the imbedded file (as well as files it might imbed).

Instead, use the .EC [Execute Control] control word to have the input line treated as a control word even though a macro of the same name might be defined. For example, the following lines:

```
.dm im on
.ty Imbedding &* . . .
.ec .im &*
.dm off
```

redefine the .IM [Imbed] control word, preventing an endless loop while still allowing for macro substitution in the imbedded file.

Using Symbols and Macros as Associative Memory

When your document contains a large number of figures, updating the document with a new figure might mean that you have to renumber all subsequent figures. Manually, it is a time-consuming and error-prone task.

With symbols, SCRIPT/VS can automatically keep track of the numbering you need and can provide more convenient figure referencing. You can also build a list of figures, including figure numbers and page numbers, automatically. Most important, you can rearrange the figures as often as you please without having the monumental task of renumbering the figures and their references each time.

To number figures, use a *counter*, which is a unique symbol name that refers to (and contains the value of) the current figure number. The figure number symbol is set at the beginning of the input file or in a separate file that is imbedded at the beginning of the input file. To manage the counter, define macros for figures and figure references in the profile of your document:

```
.se figctr = 0
.*
.dm fignum on
.se figctr = &figctr + 1
.se fig@&*1 = &figctr
.se fig#&*1 = &
Figure &fig@&*1..
.dm off
.*
.dm figref /Figure &@&*1 on page &#&*1
.su on
```

Whenever you enter a figure in your document, invoke the FIGNUM macro with a unique *identifier* just before the figure caption:

```
.fl on
(body of figure)
.fignum fred
Example of Aardvark's Table Manners
.fl off
```

The FIGNUM macro assigns the figure the unique identifier *fred* and does the following:

- Increments the figure counter
- Saves the number of the fred figure in the symbol &fig@fred
- Saves the page number of the fred figure in the symbol &fig#fred
- Inserts the word *Figure* and the figure number in front of the figure caption.

Whenever you want to refer to the figure you have called “fred” in the text of your document, use the FIGREF macro:

```
... as shown in
.figref fred
```

The FIGREF macro inserts a string containing the appropriate figure number and page number into your document:

```
... as shown in Figure 4 on page 123 ...
```

To automatically build a list of illustrations, the FIGNUM macro could have been defined like this:

```
.se figctr = 0
.*
.dm fignum on
.se figctr = &figctr + 1
.se *sx '?Figure &fig@&*1..? .?&fig#&*1..?'
.dm figlist() |.sx &*sx
.se fig@&*1 = &figctr
.se fig#&*1 = &
Figure &fig@&*1..
.dm off
.*
.dm figref /Figure &@&*1 on page &#&*1
.su on
```

Notice that the FIGLIST macro is actually defined by the FIGNUM macro. You can use the other forms of the .DM control word within an inline form of .DM, but an inline form of .DM within an inline form will cause the initial macro definition to end.

At the end of the first pass, the FIGLIST macro will contain one line for each figure in the document, and each line will consist of a .SX [Split Text] control word that will format a figure number and page number.

Note: The lines of the FIGNUM macro that build the FIGLIST macro appear before the lines of the FIGNUM macro that set the symbols referred to in the FIGLIST macro. This is deliberate. Figures are usually enclosed in floats or keeps, and the page on which they will be placed is not known when the figure is formatted. For this reason, SCRIPT/VS processes .SE [Set Symbol] control words that refer to the page number symbol twice, once when first encountered and again when the page on which surrounding text will be formatted is known. To ensure that the page numbers in the list of illustrations are correct, SCRIPT/VS delays substitution of these symbols until the FIGLIST macro is executed, when all figures have been placed.

Redefining SCRIPT/VS Formatting Conventions

SCRIPT/VS has several implicit formatting functions. Input lines that are null reset line continuation, and those that begin with a blank or tab character cause a break. You can use a macro to redefine these functions.

Processing Input Lines That Begin with a Blank or a Tab

When an input line begins with a blank (called a leading blank) or a tab (called a leading tab), SCRIPT/VS does not concatenate the line with the previous input line. That is, a break occurs.

Breaks are provided by processing the .LB [Leading Blank] control word when a leading blank is encountered, and by processing the .LT [Leading Tab] control word when a leading tab is encountered. Both of these control words function exactly the same as the .BR [Break] control word. However, after the break occurs, the leading blank or tab remains on the input line and is processed as part of the line.

You can control the actions to be taken for leading blanks and tabs by defining a .LB and .LT macro. When you want the leading blank and leading tab to be processed by SCRIPT/VS as just a blank (or just a tab) that happens to occur as the first character (that is, not processed differently than other blanks or tabs), redefine the control words as:

```
.dm lb /.  
.dm lt /.
```

The blank or tab at the beginning of the input line will be concatenated with the previous input line. It will not necessarily appear at the beginning of an output line.

Note: The .NL [Null Line], .LB [Leading Blank], .LT [Leading Tab], and .BL [Blank Line] functions are not performed for a line that would otherwise call for them when the line is processed in literal mode (that is, preceded by the .LI [Literal] control word). Null text lines still reset line continuation if the previous line ended with a continuation character, but the .NL control word or macro is not processed.

Specifying a Macro Library

LIB: Specify Symbol and Macro Libraries

The LIB option is valid in the CMS, TSO, and ATMS-III environments and specifies that SCRIPT/VS is allowed to search the specified libraries for a definition of the symbols and macros not defined within the input file. In a batch environment, the SEARCH option provides a similar facility.

In CMS, the LIB option is specified as:

```
LIB (libname1 [ ... libname8 ] )
```

where *libname* is the filename of a CMS macro library. The filetype is MACLIB. The CMS search sequence is used to locate the library on any accessed disk.

In TSO, the LIB option is specified as:

```
LIB (libname)
```

If the *libname* given is not fully qualified (placed within quotation marks), the userid is prefixed to the *libname* as the leftmost qualifier, and *MACLIB* is added (unless it already appears) as the rightmost qualifier.

In ATMS-III, the LIB option is specified as:

```
LIB (opnum1 [ ... opnum8 ] )
```

where *opnum* is an operator number. It must include the user's number if the user's permanent storage is to be searched.

The library is searched when a symbol or macro is not already known and SCRIPT/VS has encountered a .LY ON, a .LY SYM (only for symbols), or a .LY MAC (only for macros) control word. The library is also searched (without regard to the setting of the .LY control word) when a symbol or macro is defined with the LIB parameter. For example:

```
.se symbolname LIB
```

```
.dm macroname LIB
```

If the symbol name or macro name is not found in the symbol table (but the symbol or macro is defined as being in a library), SCRIPT/VS scans each library named in the LIB option (in the order given) until the symbol or macro is found. SCRIPT/VS then moves the symbol or macro definition into the SCRIPT/VS symbol table, so that a second occurrence of the macro or symbol does not require a library search. If no LIB option is specified, the symbol name or macro is searched for in the default library (if it exists).

When a macro name cannot be resolved (because there was no previous definition set with a .DM [Define Macro] control word), SCRIPT/VS can look for its definition in a macro library.

The member name of each macro defined in the macro library is the macro name without the leading period. It is restricted to eight characters. Symbol definitions and macro definitions can be members of the same library. Only the first line of a macro library member is read for a symbol definition; for a macro definition, all lines of the member are read and treated as individual lines of the macro definition.

You can use the macro library in two ways:

- To explicitly set a macro name. Use the LIB parameter of the .DM [Define Macro] control word to instruct SCRIPT/VS to retrieve its definition from a library:

```
.dm para lib
```

SCRIPT/VS searches the library specified by the LIB option of the SCRIPT command for the definition of .PARA and retrieves the definition. The retrieved definition replaces any existing definition.

- To define an unresolved macro. When SCRIPT/VS encounters a macro that has not been defined, the library is searched for a member with the same name as the macro when either .LY ON or .LY MAC has been specified.

When your input file contains macros that are defined in a macro library, specify either .LY [Library] ON or .LY [Library] MAC to instruct SCRIPT/VS to search the macro library for any unresolved macro it encounters:

```
.ly on      or      .ly mac
```

The ON parameter specifies that the macro library is to be searched for unresolved macros and symbols. The MAC parameter specifies that the macro library is to be searched only for unresolved macros. You can use the OFF or SYM parameters of the .LY control word to turn off library searching for unresolved macros.

Because searching macro libraries for unresolved symbols is expensive in terms of processing time, we recommend that .LY MAC be used except for short periods when you expect symbol definitions to be returned; then .LY SYM or .LY ON should be used.

In ATMS-III, the search technique is the same for both symbols and macros. Therefore, it does not matter whether .LY MAC, .LY SYM, or .LY ON is used.

Creating SCRIPT/VS Macro Libraries

Macros that are going to be used for multiple documents can be stored in a macro library. How you create your macro libraries is determined by the environment in which you are operating SCRIPT/VS.

When you are placing the definition of a macro into a macro library, ensure that none of the lines of the macro is preceded by a .DM [Define Macro] control word.

In a CMS Environment

In a CMS environment, a SCRIPT/VS macro library must have a file type of MACLIB. Members can be edited directly using SPF/CMS, but not with the CMS editor or the Display Editing System.

A macro can be created or changed by editing a file with a file name that is the same as the macro name and a filetype of COPY. The record format of the file must be fixed, and the record length must be 80 bytes.

The CMS MACLIB command is used to add or replace macros in a macro library. To modify an existing macro, you must have the text of the macro punched to your virtual card reader, and then read into a COPY file. This makes the macro accessible to the CMS editor. (This procedure is described in detail in the *Virtual Machine Facility/SP: CMS User's Guide*.)

You can use the LIB option of the SCRIPT command to specify as many as eight macro library names.

```
script test ( lib (mylib yourlib)
```

The file type for all of these libraries must be MACLIB. When an undefined macro is encountered, the libraries will be searched in the order specified on the LIB option of the SCRIPT command. If no library name is specified using the LIB option, a default name of DSMGML4 MACLIB is used.

In a TSO Environment

In an interactive or batch TSO environment, SCRIPT/VS macro libraries must be cataloged and must be defined as partitioned data sets (DCB=DSORG=PO). SCRIPT/VS does not establish default data set attributes for macro libraries. Therefore, it is especially important when creating a library that the correct data set attributes are used. SCRIPT/VS macro libraries must meet all of the requirements outlined in "Characteristics of an Input File" in the *DCF: SCRIPT/VS Language Reference*. For best performance, the data set should not contain line numbers. If record format fixed (DCB=RECFM=F) or fixed block (DCB=RECFM=FB) is used, a logical record length of 80 bytes is suggested (DCB=LRECL=80).

For variable or fixed-format data sets, a block size should be selected that is compatible with the logical record length and the characteristics of the physical devices where the macro library will reside. A standard block size and record format should be used for all SCRIPT/VS macro libraries within an installation. Unpredictable errors can occur if macro libraries are concatenated that have different record formats, or if the first data set in a concatenation sequence does not have a block size that is equal to or larger than the block size of every other library in the concatenation group.

The default first and last data set name qualifiers for a SCRIPT/VS macro library are the TSO data set name prefix qualifier (the USERID by default) and MACLIB, respectively. The correct choice of a data set name for a macro library can reduce the amount of typing required to enter the SCRIPT/VS command. For more information about data set name qualification, refer to “MVS/TSO Data Set Naming Conventions” and “LIB: Specify Symbol and Macro Libraries” in the *DCF: SCRIPT/VS Language Reference*.

The default name of the SCRIPT/VS macro library is SCRIPT.R40.MACLIB; however, this name can be changed. See Chapter 10, “Tailoring SCRIPT/VS for Your Installation” on page 111 for information on how to change the data set name at your installation.

One private library may be concatenated to the default SCRIPT/VS macro library, using the LIB option of the SCRIPT command. Multiple macro libraries may be concatenated if the libraries are preallocated with a DDname of SCRIPTLIB. For more information on the concatenation of SCRIPT/VS macro libraries, refer to “LIB: Specify Symbol and Macro Libraries” in the *DCF: SCRIPT/VS Language Reference*. Preallocation of a SCRIPTLIB DDname may be accomplished with the TSO ALLOCATE command or through the TSO LOGON procedure JCL. The TSO ALLOCATE command is documented in the *VS2 TSO Command Language Reference* manual, and information about TSO LOGON procedures may be found in *OS/VS2 MVS System Programming Library: TSO*.

Note: No matter what method is chosen to access the macro libraries, the libraries must be cataloged under the real data set name. SCRIPT/VS does not support data set name aliases.

Members of a macro library can be added or changed directly using either the TSO system editor or the Structured Programming Facility-II (SPF-II) editor. The SPF-II utility function can be used to delete members or to list member names. Because changing or deleting members leaves nonreusable free space within a macro library, libraries should be reorganized after updating.

In an ATMS-III Environment

In an ATMS-III environment, macros can be created as individual documents. These individual documents cannot cover other customer-written applications. However, they can be accessed as either documents or subdocuments. Macros that are accessed as individual documents must reside in permanent storage and must have uppercase names. Macros that are accessed as subdocuments must also reside in permanent storage, but need not have uppercase names.

In ATMS-III, the LIB search is used only if the requested source cannot be located through the use of ATMS subdocument index build or connect facilities. Furthermore, the search is performed only against the permanent storage of the users whose operator numbers are specified in the LIB list.

If other operators are going to be using your macros, you must store them with a getword of *any*. Another operator can then access them by specifying the LIB option and your operator's number. If you are going to use macros that are stored in your permanent storage and macros that are stored in another operator's permanent storage, you must specify both your number and the other operator's number when specifying the LIB option.

```
script * ( lib (myopnum youropnum)
```

If all of your macros, symbols, and GML tags are going to be accessed as subdocuments and they are all stored in your permanent storage area, you do not have to specify the LIB option. Refer to the *ATMS-III Terminal Operator's Guide* for more information on creating and using subdocuments.

If the LIB option is not specified, ATMS-III uses only its subdocument facilities to search for unresolved symbols and macros.

When you are going to access macros as subdocuments, you might want to give them names with a common prefix. This enables you to build and connect them based on the common prefix. For example, if you created these macros:

```
testPARA
testKEEP
prodPARA
prodKEEP
```

issuing these ATMS-III commands:

```
build;m;test
connect;x;test
```

results in the macros .PARA and .KEEP being retrieved from the documents testPARA and testKEEP, rather than prodPARA and prodKEEP.

| In a DLF Environment

| In a batch environment, DLF can be used to invoke SCRIPT/VS. Therefore, any macro libraries that are required for processing a document must be created as sequential data sets and brought into the Document Library Facility's Document Library by the IMPORT command before they can be accessed by SCRIPT/VS. For more information on how to use the IMPORT command to bring documents into the Document Library using the Document Library Facility, and how to access documents stored in the Document Library, refer to the *Document Library Facility Guide*.

Chapter 7. Processing GML

Chapter 7. Processing GML	71
GML Markup Syntax	72
Changing the GML Delimiters	73
SCRIPT/VS Processing of GML	74
Automatic GML Processing	74
Attribute Scanning Rules	75
Attribute Processing	76
Value Attribute Processing	77
Residual Text Processing	77
GML Tag-to-APF Mapping	79
Explicit Mapping	79
Class Mapping	79
Direct Mapping	79
Creating Your Own GML Tag	80

Chapter 7. Processing GML

GML can be used to describe the structure and elements of your document without regard to the particular processing that can be required. Like other languages, GML has a syntax and usage rules, but GML has no fixed vocabulary. You can develop your own vocabulary of tags to describe your documents. The DCF actually provides two languages: SCRIPT/VS formatting language and GML descriptive language. One way to characterize the difference between the two languages is this: the formatting language is made up, basically, of verbs that indicate what processing to perform; on the other hand, GML is made up, basically, of adjectives that describe the structure and elements of a document.

DCF also provides profiles and macro libraries to support sets of tags for foils, general documents, memos, and schedules. You can use these sets of tags as examples of how to implement GML, or you can use these sets of tags, where appropriate, and add your own tags to tailor the GML vocabulary to describe your documents. Refer to the *DCF: GML Starter Set Implementation Guide* for more details on these sets of tags.

The GML functions of SCRIPT/VS are enabled with the .GS [GML Services] control word:

```
.gs tag on
```

The profiles provided with GML execute this control word.

Note: The *unmodified* GML profiles and macro libraries are a supported component of DCF.

The GML profiles require certain fonts determined by the device you are using.

Table 2. Required Fonts	
Device	Required Font
3800 Model 3 and 3820	5771-ABA Sonoran Serif 5771-ABC Pi and Specials The Prestige Typeface family included with the Print Services Facility and Print Management Facility licensed programs.
PostScript devices	Courier and Times Roman typeface families A font index file and the Adobe Font Metrics files for the supported fonts are shipped with DCF.
4028	Courier and Times Roman typeface families These fonts are available in PSF Version 2 (MVS and VM).
4224	Courier and Essay typeface families These fonts are included with PSF.
4250	5771-AAR Monotype Times New Roman 5771-AAW Typewriter and Pi Specials

GML Markup Syntax

GML tags can appear anywhere in an input document and are identified by the GML delimiter, which, by default, is a colon (:). A control word should never precede a tag in the same input line. If doing so is absolutely necessary, then use the control word separator symbol (&\$CW) instead of the control word separator character. A GML tag name can be up to 8 characters long and can consist of letters, numbers, and the characters @, #, and \$ (except that the first character cannot be numeric). The tag name can be entered in either uppercase or lowercase. For example, in the GML starter set provided with SCRIPT/VS, the following tag identifies a place where a list of illustrations should be generated:

```
:figlist
```

This same tag can also be entered as:

```
:FIGLIST
```

GML tags indicate where specific document elements begin. Some elements also require an explicit *end-tag* to indicate the end of the element. GML end-tags are identified by the GML end-tag delimiter, by default a double colon (::), and have the same naming rules as GML tags. For example, an ordered list might be indicated as:

```
:ol  
.  
.  
.  
:  
::ol
```

Some GML tags recognize *attributes*, which further describe the document element identified by the tag. Attributes follow the tag name, separated by one or more blanks, and have the same naming rules as GML tags. Attributes also have values, which follow the attribute name, separated by an equal sign:

```
:fig frame=box
```

When an attribute value contains blanks or special characters, it must be enclosed in single quotation marks ('):

```
:gdoc sec='Company Confidential'
```

If the value itself contains quotation marks, they should be doubled.

```
:h1 stitle='Programmer's Guide'.DCF: Text Programmer's Guide
```

Some tags recognize attributes that consist of a single word. These are called *value attributes*. They are entered just as other attributes, but without an equal sign:

```
:ol compact
```

Whenever text follows markup, the text should be delimited by a *markup/content separator* (MCS), which is by default a period (.). For example:

```
:p.While there's no cause for alarm,  
there is no room for complacency.
```

The line of text following the markup/content separator is the *residual text* for the GML tag. In the example above, the residual text for the *p* tag is "While there's no cause for alarm."

The residual text can be null, if no text appears between the end of markup and the next tag. If there is text between the end of markup and the next tag, then the residual text is the first line of text. For example:

```
:ol
:li.
A solitary list item.
::ol
```

The residual text for the *ol* tag is null, whereas the residual text for the *li* tag is "A solitary list item."

Normally, residual text is formatted along with any other text following the markup. See "Residual Text Processing" on page 77 for more details on the treatment of residual text.

The markup/content separator need not be entered if:

- The markup is immediately followed by another tag
- Whatever follows the markup cannot be misconstrued as an attribute.

The markup/content separator can appear anywhere on an input line; however, if the MCS character is the period (.), SCRIPT/VS will interpret it as a control word delimiter if it appears in the first character position of an input line. You should, therefore, avoid starting a line with a markup/content separator.

GML markup can span as many lines in the input document as necessary, and blanks between attributes are ignored. For example, a tag, its attributes, and the residual text can all be entered on a single line:

```
:h1 id=gml.GML Support in SCRIPT/VS
```

or a tag, each of its attributes, and the residual text can all be entered on separate lines:

```
:h1
  id = gml
  stitle= 'GML Support'
Generalized Markup Language Support in SCRIPT/VS
```

Each input line can have one or more attributes on it, separated by one or more blanks, but each attribute must be entirely contained on a single line. The markup can end on any line, with the residual text line following all on that same line or all on the next line.

Note: GML markup cannot span input files.

GML scanning may be ended by another tag, by a control word at the start of an input line, or by an end-of-input file condition.

Changing the GML Delimiters

The GML tag and end-tag delimiters and the markup/content separator can be changed with the .DC [Define Character] control word.

The GML tag delimiter can be set to any character that is not valid in a tag name, except ampersand (&). For example:

```
.dc gml !
```

With this delimiter, the list of illustrations would be identified as:

```
!figlist
```

The GML end-tag delimiter can be one or two characters. If it is a single character, it can be any character that is not valid in a tag name, except ampersand (&) and the GML tag delimiter. For example:

```
.dc gml $ ¢
```

With these delimiters, an ordered list would be identified as:

```
$ol  
.  
.  
.  
¢ol
```

If the GML end-tag delimiter is two characters, the first must be the same as the GML tag delimiter. For example, in the GML starter set the delimiters are set as:

```
.dc gml : : e
```

With these delimiters, an ordered list would be identified as:

```
:ol  
.  
.  
.  
:eol
```

The markup/content separator can be set to any character that is not valid in a tag name, except ampersand (&). For example:

```
.dc gml <  
.dc mcs >
```

With these delimiters, tags can be entered as:

```
<h1 id=gml>GML Support in SCRIPT/VS
```

SCRIPT/VS Processing of GML

This section describes the functions available in SCRIPT/VS to recognize GML markup and associate the tags and attributes with APFs.

Automatic GML Processing

When SCRIPT/VS processes a document and encounters a GML tag, the following processing sequence occurs:

1. Any attributes not processed by the previous tag are purged.
2. A search is made for an application processing function (APF) for the tag. This APF (written in the SCRIPT/VS formatting language and usually a macro) may be:
 - An APF associated with the tag by the .AA [Associate APF] control word
 - An APF with the same name as the tag.
3. The input is scanned for attributes and value attributes, if recognized by the tag, and they are saved for processing.

4. The residual text line is identified and saved.
5. The APF is invoked.
6. Any residual text is processed.

See Figure 1 for an illustration of how SCRIPT/VS processes documents containing GML tags.

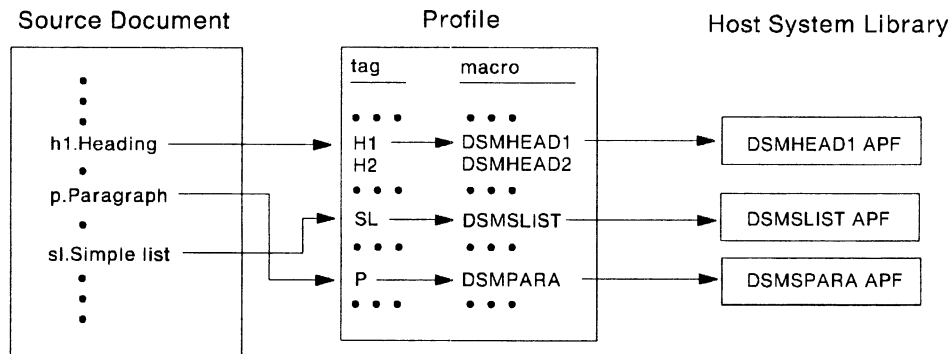


Figure 1. Processing Documents with GML. The profile provides the mapping between tags, which identify elements of text in the source document, and APFs, which provide formatting functions.

Attribute Scanning Rules

The `.GS [GML Services] RULES` control word can be used to specify:

- Whether attributes are allowed for tags
- Whether value attributes are allowed
- What to do if an invalid attribute is found:
 - Stop the scan and treat the invalid attribute as text
 - Skip the invalid attribute and keep scanning.
- Whether to issue a message if an invalid attribute is found or whether to take corrective action.

Attribute scanning rules can be specified separately for GML tags and end-tags. For example, in the GML starter set provided with SCRIPT/VS:

```
.gs rules (att novat stop nomsg) (noatt)
```

specifies that GML tags can have attributes but not value attributes, that attribute scanning should stop without a warning message when an invalid attribute is found, and that GML end-tags will not recognize attributes. Invalid attributes are most commonly text, encountered when an optional markup/content separator has been omitted.

The attribute scanning rules for tags given with `.GS RULES` can be overridden for specific tags with the `.AA [Associate APF]` control word. For example, in the GML starter set:

```
.aa ol dsmolist (vat) dsmolist
```

indicates that the APF for the `ol` tag is the `DSMOLIST` macro, and that value attributes are allowed for this tag.

The APF for the *ol* end-tag is the DSMELIST macro, and because no attribute scanning rules are specified, those given with .GS RULES for end-tags will be used.

The attribute scanning rules for .AA and .GS RULES are described in the SCRIPT/VS control word description section of the *DCF: SCRIPT/VS Language Reference*.

Attribute Processing

For example, within the APF for a tag:

```
:fig id=fred place=inline frame=box
```

the .GS [GML Services] EXATT control word, specified as:

```
.gs exatt
```

can be used to process all of the attributes that have been found and placed in the attribute stack.

The attributes are processed by APFs of the same names as the attributes. In other words, id is processed by the ID APF, place is processed by the PLACE APF, and frame is processed by the FRAME APF.

It is also possible to selectively process attributes. If the APF that processes the fig tag specifies:

```
.gs exatt frame id as @idf
```

then the frame attribute is processed by the FRAME APF and the id attribute is processed by the .@IDF APF. The place attribute is not processed unless specified on another .gs exatt control word line.

The value of the attributes is provided to the APFs as parameters. For example, box is provided to the FRAME APF as the parameter &*1, and the value fred is provided to @IDF as &*1.

The APF for the fig tag can also specify:

```
.gs exatt width
```

but because the width attribute was not specified with the tag, the width macro is not processed.

Another way to process attribute names is by specifying the ATTVAL parameter on the .GS [GML Services] control word.

Suppose the following GML tag is entered:

```
:tagname id='fred' place='des moines'.
```

Then the APF for 'tagname' would contain:

```
.gs attval id place
```

which sets the symbol &ID to "fred" specified in the ID = attribute, and it sets the symbol &PLACE to "des moines". Note that the symbol name is always upper-case.

If you want to set a different symbol, use the AS keyword:

```
.gs attval id as *xyz
```

This sets the macro local symbol &*xyz to the value of the ID= attribute (in this case, "fred"). If you specify a symbol name with the AS keyword, it is not folded to uppercase; the name is used exactly as you specified it.

Attribute processing is described in further detail in the description of the .GS [GML Services] control word in the *DCF: SCRIPT/VS Language Reference*.

Value Attribute Processing

Value attributes are presented to the APF for the tag as the parameters &*1, &*2, and so on. The number of value attributes is provided in &*0.

For example, if a simple list tag were encountered:

```
:sl compact.
```

the value compact would be provided in &*1 to the APF that processes the SL tag. The APF could test whether this symbol was compact and then proceed accordingly.

In the following example:

```
.if /&U'&*1 eq /COMPACT  
.th (do something)  
.el (do something different)
```

the &U' symbol attribute was used to ensure that the uppercase form of &*1 would be used for comparisons because we do not know whether it was entered in upper- or lowercase. The symbol &*1 is a null symbol if the COMPACT attribute is not specified, and it will cause an error if it is used alone on the IF control word, so it is prefixed here with a slash (/).

Residual Text Processing

For many elements, the APF operates by setting up the correct formatting environment and then allowing the following text to be formatted under the control of this environment. In these cases, the APF does not need to process the residual text line directly; SCRIPT/VS automatically retrieves the residual line and processes it after the APF has completed its function. SCRIPT/VS automatically provides continuation, if necessary, so that if the GML markup occurred in the middle of a word, the processing (such as starting a new font, for example) will not break the word. Because of this continuation, if the user is running with the SPELLCHK option, the first word of residual text may not be spellchecked.

If the APF needs to process the residual line directly, the APF can retrieve the residual line with the .GS [GML Services] SCAN control word:

```
.gs scan line
```

The residual text is removed from the document and is placed in the symbol &line. If you do not want the residual text to be removed from the document but only want to have a copy of it placed in the symbol &line, you can specify:

```
.gs copy line
```

When an APF explicitly retrieves the residual text, it is the APF's responsibility to provide continuation or other special treatment that may be required, such as turning on literal mode for the residual piece.

Residual text is treated as literal text; that is, special processing, such as execution of another control word, is not performed if the line begins with a leading blank, tab, or control word separator. Normally, residual text is formatted along with any text following the markup; however, in format off mode, a tag in the middle of an input line may cause two or more output lines if that tag contains control words that cause a break.

GML scanning was designed to enable the creation of APFs and tags that can be used to describe the structure and elements of your documents. It was not designed as a means of introducing text, such as “boilerplate phrases.” If such text is deemed necessary, you should avoid, if possible, having the text at the end of the APF. If this is not possible, then use a continuation character at the end of the phrase. For example, the lines:

```
.gs tag on
.dm text /phrase
:text.,
xxx :text.,
```

will result in:

```
phrase , xxx phrase, (notice the blank before the first comma)
```

whereas:

```
.dc cont +
.gs tag on
.dm text /phrase+
:text.,
xxx :text.,
```

will result in:

```
phrase, xxx phrase,
```

If you want to insert text from an APF, use the INSERT parameter of the .GS control word. For example, if you specify:

```
.gs insert Figure 7
```

then the text, Figure 7, will be inserted, using proper continuation, before any residual text that may exist for the tag associated with this APF.

The continuation of text that comes before and after a GML tag can be affected by other markup on the input line. For example, if you entered:

```
.sp; text1 :tag text2
```

then the .SP [Space] control word causes a break between *text1* and *text2*. The resulting output may appear as:

```
text1
text2
```

Any control words entered at the beginning of an input line that contains both text and GML tags will have this effect.

GML Tag-to-APF Mapping

GML scanning is enabled with the .GS [GML Services] control word:

```
.gs tag on
```

When a valid GML tag is found, SCRIPT/VS attempts to locate an APF for the tag. The APF, which can be a macro or a control word, can be found by:

- Explicit mapping (established with the .AA [Associate APF] control word)
- Class mapping (established with the .GS [GML Services] PREFIX control word)
- Direct APF mapping (a macro or control word with the same name as the tag).

If no APF is found, a warning message is issued, and the tag is treated as text. If you do not want to be warned about invalid tags, specify:

```
.gs tag onno
```

Explicit Mapping

The .AA [Associate APF] control word allows you to explicitly specify the APFs for particular GML tags and end-tags. For example, to define tag-to-APF mappings for the *fig* tag and end-tag, specify:

```
.aa fig figure figurex
```

The APF for the *fig* tag is the FIGURE macro, and the APF for the *fig* end-tag is the FIGUREX macro.

The .AA control word also allows you to specify the attribute scanning rules for each tag, as described under “Attribute Scanning Rules” on page 75. The .AA control word is described in more detail in the *DCF: SCRIPT/VS Language Reference*.

Class Mapping

A single character that will be added to the front of a tag name to produce an APF name can be specified with the .GS [GML Services] PREFIX control word. For example:

```
.gs prefix @
```

With this class mapping in effect, the APF for the *figlist* tag is the @FIGLIST macro.

Direct Mapping

If no other tag-to-APF mapping is provided for a tag, a macro or control word whose name matches the tag name is used as the APF. This is the default.

Creating Your Own GML Tag

Before you create a tag of your own, you will need to:

- Identify the need for a tag
- Decide on a tag name
- Identify the tag's function
- Define an APF to process the tag
- Enable the tag.

Suppose, then, that you need a frequently used disclaimer that contains some constant and some variable text.

**Any Similarity To Living Or Dead Persons
Of The Same Name Is Purely Coincidental.**

For Legal Questions Contact: John Doe

In our example, the variable text is the name of the person to contact, the rest of the disclaimer will always be the same.

Your first step would be to give names to your tags.

- `:disc` - to start the structure
- `:edisc` - to end the structure.

The markup might look like this:

```
:disc.John Doe  
:edisc.
```

Your second step is to define the APF for the `:disc` tag. This APF will:

- Get to the top of a page
- Turn on highlighting
- Start a box
- Insert some space
- Left align the text of the message
- Print the text of the message

It might look like this:

```
.dm lnote on
.pa nostart
.bf hi2
.bx 1 45
.sp 1
.in +2
.ir +2
.fo left
Any Similarity To Living Or Dead Persons
Of The Same Name Is Purely Coincidental.
.sp 1
For Legal Questions Contact:&rbl.&$cont.
.dm off
```

Note that the two symbols in the last line of our disclaimer text (&rbl. and &\$cont.) are there to put a required blank between the text and the variable name to follow, to keep that variable name and the text on the same line.

Then the APF for the end tag should:

- Turn on normal formatting
- Leave some more space
- Complete the box
- Turn off highlighting.

It might look like this:

```
.dm elnote on
.fo
.sp 1
.bx off
.pf
.dm off
```

Your last step would be to integrate the tags into your document or into a profile that you created that then imbeds DSMPROF4 (the profile for the GML starter set that, among other things, turns on tag and macro processing, enables the macro library, and associates GML tags that you have enabled with the .AA [Associate APF] control word to their correct APFs). In the example, you would associate the tags :disc and :edisc like this:

```
.aa disc lnote elnote
```

Your complete tag description might look like this:

```
.  
.*  
.dm lnote on  
.pa nostart  
.bf hi2  
.bx 1 45  
.sp 1  
.in +2  
.ir +2  
.fo left  
Any Similarity To Living Or Dead Persons  
Of The Same Name Is Purely Coincidental.  
.sp 1  
For Legal Questions Contact:&rb1.&$cont.  
.dm off  
.*  
.dm elnote on  
.fo  
.sp 1  
.bx off  
.pf  
.dm off  
.*  
.aa disc lnote elnote
```

With your tag complete, you can now specify:

```
:disc.John Doe  
:edisc.
```

and you will get the disclaimer:

**Any Similarity To Living Or Dead Persons
Of The Same Name Is Purely Coincidental.**

For Legal Questions Contact: John Doe

printed at the top of your next page with the variable text (in this case, John Doe) added as a result of the processing of your tag.

Chapter 8. Verifying Spelling

Chapter 8. Verifying Spelling	85
Spelling Verification	85
Spelling Fallibility	88
The SCRIPT/VS Dictionaries	89
Building a User Dictionary	91
Building an Addenda Dictionary	91
TLIB: Specify Spelling Checking and Hyphenation Libraries	92
Searching a SCRIPT/VS Dictionary	93
Stem Processing	93
English Prefixes and Suffixes	94
French National and Canadian French Prefixes and Suffixes	95
Dutch Prefixes and Suffixes	96
Italian Prefixes and Suffixes	97
German Prefixes	99
Spanish Prefixes	101

Chapter 8. Verifying Spelling

SCRIPT/VS can automatically verify the spelling of words. When this function is activated, each word in your document will be checked for correct spelling. The SCRIPT/VS dictionaries, described later in this chapter, are used for spelling verification and hyphenation.

Spelling Verification

The spelling of words in your input file will be checked by the SCRIPT/VS spelling verification function when you include the SPELLCHK option in the SCRIPT command.

The SPELLCHK option causes SCRIPT/VS to verify spelling. Each word is verified using the spelling and hyphenation dictionaries specified with the .DL [Dictionary List] control word, unless spelling verification has been turned off with the .SV [Spelling Verification] control word. If no .DL control word has been encountered, the default dictionary for your installation will be used. The &SVLG system symbol indicates the current spelling verification and hyphenation dictionary language. Spelling errors are listed with other errors found during formatting, using the .UW [Unverified Word] control word.

Whenever misspelled words are found in an input line, the .UW control word is executed with the misspelled words as parameters. This control word issues an error message to tell you that those words were not verified.

However, if you wish to have some function performed when a misspelled word is encountered, you can define a .UW macro. When macro substitution is on, your .UW macro will be executed whenever misspelled words are found. Note, however, that after the .UW control word or macro is processed, the misspelled words are still on the line, and are processed as part of that text input line. In other words, you cannot use the .UW macro to correct or remove such words from a line.

When unverified words are found, you may want to add them to an addenda dictionary, using the .DU [Dictionary Update] control word, so that only the first occurrence is detected; or you may want to write the words to a file to use later as an addition to your dictionary.

The following .UW macro will do both functions:

```
.dm uw on
.mg //Unverified Words: &*
.du add &*
.wf .du add &*
.dm off
.ms on
```

After you have formatted a document containing this macro with the SPELLCHK option of the SCRIPT command, the DSMUTWTF file will contain a list of all unverified words, prefixed with .DU ADD. This file must be edited to remove any misspelled words and can then be renamed and imbedded the next time the document is formatted to create an addenda dictionary.

For purposes of spelling verification, a *word* is a string of 2 to 55 characters delimited by *word delimiters*. The WORD parameter of the .DC [Define Character] control word specifies characters that are to be interpreted as word delimiters during spelling verification. The spelling of each string marked with word-delimiter characters is separately verified. A list of the default word-delimiter characters is in Table 3.

Table 3. Characters that Delimit Words for Spelling Verification. The .DC [Define Character] control word can be used to make other characters word delimiters.					
Code	Character	Code	Character	Code	Character
05	Tab	4F		6F	?
11	Special Blank ¹	5A	!	7A	:
12	Special Blank ¹	5B	\$	7E	=
13	Special Blank ¹	5C	*	7F	"
16	Backspace	5D)	8B	{
40	Blank	5E	;	0B	}
41	Required Blank ²	5F		A0	·
4A	¢	61	/	AD	[
4B	. (Period)	6B	,	AF	•
4C	<	6C	%	BD]
4D	(6D	_		
4E	+	6E	>		
¹ Special blanks are used for justification in documents formatted for the 3800 Printing Subsystem. ² The required blank is a blank that cannot have space added to it during justification. The code point assignment of the required blank can be changed with the .DC [Define Character] control word.					

The characters given with WORD are added to the current list of characters already requested.

The following example shows how the WORD parameter is used to separate individual words for spelling verification. A hyphen (-), for instance, is initially a punctuation character, so the term *in-laws* will be processed by spelling verification as a single word. However, if you enter:

```
.dc word -
```

the hyphen will be treated as a word delimiter, and the term *in-laws* will be checked as two separate words.

The backslash (\) is initially neither a word delimiter nor a punctuation character, so the term *APL\360* will be verified as a single word. The backslash is not keyable on some terminals, but it can be identified as a word delimiter by entering its hexadecimal code on any terminal:

```
.dc word e0
```

The term *APL\360* will now be processed as two separate words.

The PUNC parameter of the .DC [Define Character] control word specifies characters that are to be recognized as punctuation for spelling checking. When punctuation characters occur within a word, they will be retained when the word is checked against the dictionary; if they occur at either end of a word, they will be removed before checking takes place.

The initial punctuation characters are the hyphen (-) and single quote ('). Characters given with PUNC will be added to the current set of pf punctuation characters if ADD is specified; they will be deleted if DEL is specified.

For example, the slash (/) is initially a word delimiter character, so when the term *SCRIPT/VS* is processed by spelling verification, *SCRIPT* and *VS* are checked separately as two different words. If you enter:

```
.dc punc /
```

the slash will now be treated as a punctuation character, and the term *SCRIPT/VS* will be checked as a single word. If you enter the term */rubbish/*, the slashes will be removed, because they occur at the ends of the word, and the word *rubbish* will be verified.

When words are verified for correct spelling, *SCRIPT/VS* will search as many active dictionaries as necessary until a verified word is found. *SCRIPT/VS* will search the addenda dictionary first, the IBM-supplied dictionary next, and then each of the active user dictionaries in the order specified on the .DL [Dictionary List] control word.

The dictionaries for the following languages are empty; therefore, do not verify spelling for these languages unless you have created your own dictionaries.

- Danish
- Finnish
- Icelandic
- Norwegian
- Portuguese
- Swedish.

SCRIPT/VS searches for words in the following order:

1. Removes prefixes and suffixes.
2. Translates to uppercase if necessary.
3. Searches for each word variant in the order listed above.

Note: Because stem processing (the process of removing the prefix and suffix of a word) is performed only after each word is translated to lowercase, all words placed in the addenda dictionary should be in lowercase if stem processing is desired. No match will be found for a lowercase occurrence of a word if that word was added to the addenda dictionary in uppercase.

Spelling verification is normally performed using the addenda and main dictionaries with stem processing. Words that contain numbers are not checked unless requested with the NUM parameter of .SV [Spelling Verification] control word.

You can specify that:

- The addenda dictionary is not to be used:
`.sv noadd`
- Stem processing is not to be performed:
`.sv nostem`
- Words that contain numbers are to be checked:
`.sv num`

Spelling verification can also be used to verify that proper names start with an initial capital letter. For example, if an entry is made in the addenda dictionary as follows:

```
.du add Teri
```

then *Teri* and *TERI* will both be correctly spelled. However, *teri* will be regarded as misspelled.

When spelling verification is performed, each occurrence of every word in the document being formatted is checked against the active dictionaries. This can result in a significant increase in the processor time required to format a document.

Often it is sufficient to perform spelling verification only twice: once, when the document is first created, to find entry errors, acronyms, and valid words that are not in the dictionaries; and, again, just before the final formatting runs, to catch any errors made while updating or revising the document.

Spelling Fallibility

SCRIPT/VS spelling verification is not infallible. A misspelled word with a suffix or prefix could possibly yield a correctly spelled word after stem processing. For example, *disbooked* (with the stem *book*), and *missteak* will both be verified after stem processing.

Also, the stem-processing algorithms do not handle all exceptions to general spelling rules used in the English language. For example, the plural of *mouse* must be explicitly added to an addenda dictionary.

Spelling verification might fail on the first word of a GML Tag's residual text because of possible continuation from a previous tag.

Spelling verification may fail for words inserted in a line by the substitution of a symbol defined with the `.DV` [Define Variable] control word. Refer to the description of the `.DV` [Define Variable] control word in the *DCF: SCRIPT/VS Language Reference* for more information.

The SCRIPT/VS Dictionaries

Three types of SCRIPT/VS dictionaries are used for hyphenation and spelling verification:

- Read-only dictionaries of words provided by IBM with SCRIPT/VS. Each contains about 10,000 words. Because suffixes and prefixes are removed from a word, if necessary, the effective dictionary size is significantly larger.

Note: The IBM-supplied dictionaries for the following languages are empty: Danish, Finnish, Icelandic, Norwegian, Portuguese, and Swedish. If you want to use dictionaries for those languages, you must create a user dictionary. See “Replacing the IBM-Supplied Dictionaries” on page 150 for a method of substituting your own dictionary for the IBM-supplied dictionary.

- User dictionaries created by your installation using the dictionary-maintenance program. These dictionaries contain words that are not in the main dictionaries but are used in most of the documents produced at your installation. These words often reflect the nature of a particular business and usually include technical terms and company abbreviations. Once created, these dictionaries are also read-only. The algorithms used in building dictionaries require that they contain more than five words and that they are relatively dissimilar.
- Addenda dictionaries you create for a specific document using the .DU [Dictionary Update] control word. Addenda dictionaries contain words that are not in the main or user-created dictionaries but are frequently used in a specific document. This type of dictionary often includes abbreviations that apply to a particular product, jargon, and the names of people and places. It is the most temporary of the three types of dictionaries, because it is rebuilt in storage every time SCRIPT/VS processes a document that requires it. An addenda dictionary can contain a maximum of 804 words.

IBM provides root word dictionaries in nine languages:

- American English
- Canadian English
- Canadian French
- Dutch
- French
- German
- Italian
- Spanish
- United Kingdom English

The unique stem-processing routine that IBM provides with each of these languages is used by all three types of SCRIPT/VS dictionaries in performing hyphenation and spelling verification in a given language.

Use the .DL [Dictionary List] control word to specify which language you want to use for hyphenation and spelling verification. This control word automatically activates the corresponding stem-processing routine for that language, as well as any user dictionaries that are associated with that root word dictionary. If a dictionary is not specified, the default dictionary for your installation is used.

The hexadecimal code points for accented characters in the SCRIPT/VS spelling checking and hyphenation dictionaries are listed in Table 4.

If your installation does not use these hexadecimal code points, you can modify SCRIPT/VS to translate your accented character hexadecimal code points to those in the table so that spelling verification and hyphenation of words containing accented characters is done correctly. See "DSMTRNLT: National Language Character Definitions" on page 144 for information about how to make the above translations.

Dutch UC/lc	Canadian French UC/lc	French UC/lc	German UC/lc	Italian UC/lc	Spanish UC/lc	Character Name
/45 /44 /42 63/43	64/44 62/42	/44 /42	63/43	/44	/45	"A" Acute "A" Grave "A" Circumflex "A" Diaeresis (Umlaut)
/48	68/48	/48				"C" Cedilla
/51 /54 /52 73/53	71/51 74/54 72/52 73/53	/51 /54 /52 /53		/51 /54	/51	"E" Acute "E" Grave "E" Circumflex "E" Diaeresis (Umlaut)
77/57	76/56 77/57	/56 /57			/55	"I" Acute "I" Grave "I" Circumflex "I" Diaeresis (Umlaut)
					69/49	"N" Tilde
/CE /CD /CB EC/CC	EB/CB EE/DF	/CB EE/DF	EC/CC	/CD	/CE	"O" Acute "O" Grave "O" Circumflex "O" Diaeresis (Umlaut) "OE" Digraph
FC/DC	FD/DD FB/DB FC/DC	/DD /DB /DC	FC/DC	/DD	/DE /DC	"U" Acute "U" Grave "U" Circumflex "U" Diaeresis (Umlaut)
			59			Ess zet
Note: The dictionaries for Danish, Finnish, Icelandic, Norwegian, Portuguese, and Swedish are empty and do not appear in this table.						

Table 4. Code Point Assignments for Accented Characters. Accented characters in the SCRIPT/VS Spelling Verification and Hyphenation dictionaries are represented using hexadecimal code points shown under each language for uppercase (UC) and lowercase (lc) characters.

Building a User Dictionary

A user dictionary is created using the dictionary maintenance procedures that are described in Chapter 11, “Creating and Maintaining User Dictionaries” on page 157. The words that are to be placed into the user dictionary are submitted, with the appropriate JCL, to run as a batch job in a background environment.

The input record for each job is 80 bytes long and includes:

- The appropriate hyphenation for the word
- The date on which the word was placed in the dictionary.

Once the user dictionary has been built, it must be concatenated to the main dictionary (using the .DL control word) to be accessible to SCRIPT/VS.

Because it is concatenated to the main dictionary, SCRIPT/VS treats it as part of the main dictionary. Therefore, whenever you specify, with the .HY [Hyphenate] or .SV [Spelling Verification] control words, that the main dictionary is to be used for hyphenation and spelling verification, you are automatically specifying that the user dictionary is also to be used.

Building an Addenda Dictionary

You use the .DU [Dictionary Update] control word to create an addenda dictionary. Each word specified with this control word is delimited with blanks. The word can contain lowercase and uppercase alphabetic characters, the integers 0 through 9, and punctuation characters, as defined with the .DC [Define Character] PUNC control word.

If you are building an addenda dictionary for use with multiple documents, you can create a separate file to contain the .DU [Dictionary Update] control words being used to build it and then imbed this file at the beginning of any input file that requires it.

If words being added to the addenda dictionary contain accented characters, either those accented characters must be represented by the hexadecimal code points listed in Table 4 on page 90 or the DSMACTT macro call must be changed to provide the correct translations. See “DSMTRNLT: National Language Character Definitions” on page 144 for information on how to make the translations.

When you include single hyphens in a word that you are adding to an addenda dictionary, SCRIPT/VS assumes they are potential hyphenation points. Therefore, words that normally contain hyphens (for example, upside-down) should be specified with a double-hyphen for the normally appearing hyphen. For example:

```
.du add up-side--down
```

specifies two potential hyphenation points: between *up* and *side*, and between *side* and *down*. It also specifies one normal hyphen that is always to appear: between *side* and *down*.

If a hyphenation point is searched for in a particular word, but that word was added to the addenda dictionary without a hyphenation point, no other dictionaries will be searched to find a hyphenation point. If the ALT parameter was

specified on the .HY [Hyphenate] control word, the algorithmic hyphenator will not be called to try to hyphenate the word.

Before creating an addenda dictionary, you should use the .DL [Dictionary List] control word to specify the language you are using if it is other than the default dictionary at your installation. This will associate the addenda dictionary with the main dictionary for that language. For example:

```
.dl eam
.du add Paul Ri-ver-front ec-cle-si-asti-cal
```

causes SCRIPT/VS to use the American English root word dictionary, and associate the addenda dictionary with that main dictionary. The new addenda dictionary will contain the words *Paul*, *Riverfront*, and *ecclesiastical*, which are not in the main American English dictionary.

The .DU [Dictionary Update] control word can later be used to add more words to the addenda dictionary or to delete words previously added. For example:

```
.du add Ty-pog-ra-phy
.du del Paul Ri-ver-front
```

adds the word *Typography* to the addenda dictionary and removes the words *Paul* and *Riverfront* from it.

If you specify a new language prior to specifying the .DU ADD and .DU DEL control words, the new words will be placed in the addenda dictionary associated with the new language. For example:

```
.dl germ
.du add Aus-wahl-list-en Ent-wick-lung
```

causes the German main dictionary to be used instead of the American English one, and the words *Auswahllisten* and *Entwicklung* to be added to the addenda dictionary associated with this main dictionary.

Note: More than one language can be used when processing a document. However, only one language can be active at a time.

TLIB: Specify Spelling Checking and Hyphenation Libraries

The TLIB option specifies text libraries that contain IBM-supplied root word dictionaries, user-created root word dictionaries, and stem-processing routines for use in spelling verification and hyphenation.

The TLIB option is specified as:

```
TLIB ( libname1 [ ... libname8 ] )
```

libname is the name of a CMS text library. The filetype is TXTLIB. The CMS search sequence is used to locate the library on any accessed disk.

- The specified libraries are searched when a dictionary is named in the .DL [Dictionary List] control word that is not included as part of the SCRIPT/VS load module. Both the dictionary and the stem-processing routines are loaded from the libraries.
- If the TLIB option is not specified, the library searched is DSMHYLIB TXTLIB.
- If the dictionaries used are included as part of the SCRIPT/VS load module when it is created, no library is needed.

- If you want to use a user library and the default library, you must specify both on the TLIB option.

The TLIB option is valid only in CMS.

Searching a SCRIPT/VS Dictionary

Whenever SCRIPT/VS encounters a word that needs to be hyphenated, it searches the SCRIPT/VS dictionary for the word as it appears in the input line. The associated addenda dictionaries are searched first and then, if the word is not found there, the SCRIPT/VS main dictionary with which the addenda dictionaries are associated.

Note: The IBM-supplied dictionaries for the following languages are empty: Danish, Finnish, Icelandic, Norwegian, Portuguese, and Swedish. If you want to use dictionaries for those languages you must create a user dictionary.

If no match is found in any of these dictionaries, and the word, as it appears, is all in uppercase characters, all of the letters except the first are translated to lowercase, and SCRIPT/VS again searches for the word in the addenda and main dictionaries. If no match is found, SCRIPT/VS translates all of the letters to lowercase and repeats the search.

If no match is found this time, SCRIPT/VS removes the prefix and the suffix if any, to yield the word's *root*. This form of the word is then searched for in the dictionaries. If no match is found, the word will be hyphenated using an algorithmic hyphenator, if available, unless .HY NOALG was specified.

Stem Processing

The stem-processing function attempts to generate one or more possible root words from which the input word might be derived. Suffix and prefix processing are both performed on the input word. The stem-processing function will not generate a root word, or stem, less than three characters long.

When a word's prefix is removed, the resulting stem is not changed. However, when a word's suffix is removed, the stem-processing function derives the word's stem based on the spelling rules for the language being used. For example, in English the word *churches* yields the stem *church*, and the word *flames* yields the stem *flame*.

Words may have to be processed repeatedly to remove multiple suffixes before yielding a stem. For example, the word *conceptions* would lose the two suffixes *s* and *ion* before yielding the stem *concept*.

The following descriptions summarize, by language, the prefixes and suffixes SCRIPT/VS checks for during this process.

Note: The stem-processing function is not done for the following languages: Danish, Finnish, Icelandic, Norwegian, Portuguese, and Swedish.

English Prefixes and Suffixes

SCRIPT/VS checks for the following prefixes during stem processing:

ANTI	ANY	BACK	COUNTER	CROSS	DE
DIS	DOWN	EN	FORE	IN	INTER
INTRA	KILO	MACRO	MEGA	MICRO	MILLI
MINI	MIS	MULTI	NON	OUT	OVER
PRE	PRO	RE	SEMI	SOME	SUB
SUPER	TELE	TRANS	UN	UNDER	UP

SCRIPT/VS also checks for these suffixes:

' (apostrophe)	S	ED	AL	ALLY
ING	ION			

French National and Canadian French Prefixes and Suffixes

SCRIPT/VS checks two types of French National and Canadian French prefixes during stem processing: contractions that are the result of elision processing and grammatical prefixes. The following are the contractions that SCRIPT/VS checks for:

D' (DE)	J' (JE)	L' (LE/LA)	S' (SE/SI)
N' (NE)	M' (ME)	JUSQU' (JUSQUE)	LORSQU' (LORSQUE)
QU' (QUE)	T' (TE)	PUISQU' (PUISQUE)	QUOIQU' (QUOIQUE)

SCRIPT/VS also checks for these grammatical prefixes:

INTER	ENTRE	CONTRE	TRANS	SUR	ANTI
DE(S)	EN	EM	IN	IM	RE
REM	REN	RES	REDE		

The French National and Canadian French suffixes that SCRIPT/VS checks for are:

ERAI	ERAS	ERA	ERONS	EREZ
ERONT	ERAIS	ERAIT	ERIEZ	ERIONS
ERAIENT	IRAI	IRAS	IRA	IRONS
IREZ	IRONT	IRAIS	IRAIT	IRIONS
IRAIENT	OSITION(S)	ATION(S)	ATEUR(S)	ATRICE(S)
ATIF(S)	ATIVE(S)	ATIVEMENT	IVE(S)	IVEMENT
IONS	SIONS	TIONS	INS	INT
I/NMES	I/NTES	INRENT	IS	IT
I/MES	I/TES	IRENT	US	UT
U/MES	U/TES	URENT	ISSE	ISSES
ISSENT	ISSEMENT	ELLE(S)	ELLEMENT	EUSE(S)
EUSEMENT	ENT(S)	ENTES	ENCE(S)	ANT(S)
ANTE(S)	ALE(S)	AUX	ALS	ALEMENT
E	EMENT	X	AIS	AIT
AIT(S)	IEZ	AIENT	ABLE(S)	ABLEMENT
AI	AS	A	A/MES	A/TES
ABILITE(S)	IBILITE(S)	E	S	ER(S)
IEUR	EUR	EURS	RONT	

Dutch Prefixes and Suffixes

The following Dutch prefixes are processed by SCRIPT/VS during stem processing:

AAN	AARDAPPEL	ACHTEN	ACHTER
AF	ANTI	ATOOM	AUTO
AVERIJ	BANK BOEREN	BASIS BOOM	BEDRIJFS
BELEIDS	BE	BIER BUITEN	BIJEEN
BIJ	BINNEN	BLAAS	BODEM
BOUW	BOVEN	BRUTO	BUREAU
BURGER	BURO	CLUB	COMMANDO
CONTRA	DAAR	DAK	DEPOSITO
DISCONTO	DOOR	DRAAD	DRAAI
DRIE	DRIEE:N	DRUK	EENEN
EI	EIND	EXPORT	FABRIEKS
FILM	FOTO	GAS GIRO	GE GROEI
GELD	GROND	HALF	HAVEN
HER	HOEK	HOOFD	HUIS
HOEK	HUUR	IN INCASSO	JAAR
KABEL	KALK	KANTOOR	KAPITAAL
KLUB	KOMMANDO	KONTRA	KOOP
KOSTEN	LABORATORIUM	LEIDING	LOS
MAATSCHAPPIJ	MASSA	MEDE MILIEU	MEE
MICRO	MIKRO	NA	NEGENEN
NETTO	NIEUWBOUW	NIVEAU	NIVO
OCTROOI	OKTROOI	OM	ONDER
ONT	ON	OP	OVER
PERS	PLAN	POMP	POST
PRIJS	PRODUCTIE	PRODUKTIE	PROGRAMMA
PSYCHO	RADIO	RECLAME	REKLAME
RESEARCH	RIJ	RISICO	RISIKO
SALDO SPROEI	SAMEN	SCHAKEL	SCHEEPS
SLIB	STAATS	STOF	STROOM
STUDIE	TEGEN	TELEFOON	TELEGRAAF
TELEGRAM	TENTOON	TERREIN	TERUG
TOE	TOUW	TRANSITO	TROUW
TUSSEN	UIT	VAST	VEEL
VALUTA VliegTUIG	VENDU VOORT	VER	VIEREN
VERZEKERINGS VRACHT	VIJFEN	VLOEI	VLOEISTOF
VOOR	VRACHTEN	VRIJ	WAAR
WEER	WEG	WERP	WONING
ZAND	ZEE	ZELF	ZEVENEN
ZIJ	ZINK		

SCRIPT/VS also checks for the following Dutch suffixes:

ELIJKE	LIJKE	IJE	JE	ENDE
DENDE	ENDE	KTE	PTE	BARE
LOZE	E	ERE	ELE	AGE
IE:LE	DE	INGEN	IINGEN	ELIJKEN
LIJKEN	ELIJKHEDEN	LIJKHEDEN	HEDEN	SOREN
TOREN	EREN	ELEN	BAREN	LOZEN
EN	DEN	VAN	IEN	BAAR
ER	DER	LOOS	S	ELIJKHEID
LIJKHEID	HEID	END	DEND	IEND
ING	DING	IING		

Italian Prefixes and Suffixes

SCRIPT/VS checks two types of Italian prefixes during stem processing: contractions that are formed during elision processing and grammatical prefixes. The following list summarizes the contractions that SCRIPT/VS checks for:

L'	ALL'	ANCH'	BELL'	COLL'	D'
DALL'	DEGL'	DELL'	GL'	NELL'	QUELL'
QUEST'	SULL'	UN'	NEANCH'	NESSUN'	NEINT'
QUAL'	QUALCOS'	QUALCUN'	QUAND'	QUANT'	SENZ'
C'	V'	DAGL'	S'	M'	T'
BUON'	COM'	DEV'	AGL'	SUGI'	COS
TUTT	GRAND				

The following list summarizes the grammatical prefixes that SCRIPT/VS checks for:

AUTO	ANTI	APPAR	BIS	CAPO	CENTRO
CON	CONTRO	DIS	DE	EX	EXTRA
FILO	FOTO	IN	IPER	INTER	RI
SEMI	SODDIS	SOM	SOPRA	SOS	SOTTO
SUPER	TELE	TRAS	TRAT	ULTRA	SUB
VICE	PRE	SOM	STRA	'SOVRA	'S
'INTRA	'CONTRA	'SOTT			

SCRIPT/VS processes the suffixes of Italian verbs differently from the way it processes suffixes for Italian nouns. The Italian noun suffixes that SCRIPT/VS checks for are:

A	E	I	O	AZIONE	AZIONI
ANZA	ANZE	ATORE	ATORI	ATRICE	ATRICI
ATURA	ATURE	IZIONE	IZIONI	ISTA	ETTO
ETTO	ETTI	ETTA	ETTE	HETTO	HETTI
HETTA	LINO	LINI	INA	INE	INI
NCINO	TINA	TINO	AMENTE	CAMENTE	EMENTE
AMENTO	AMENTI	IMENTO	IMENTI	LMENTE	TAMENTE
CHE(CHE)	ISSIMA	ISSIME	ISSIMI	ISSIMO
ATIVO	ATIVI	ATIVA	ATIVE	ABILE	ABILI

The Italian verb suffixes that SCRIPT/VS checks for are:

A	AI	AMMO	ANDO	ANO	ANTE
ANTI	ARE	ARONO	ASSE	ASSI	ASSIMO
ASTE	ASTI	ATA	ATE	ATI	ATO
AVA	AVAMO	AVANO	AVATE	AVI	AVO
O	INO	E	EMMO	ENDO	EI
ENTE	ENTI	ERONO	ESIMO	ESSE	ESSERO
ESSI	ESSIMO	ESTE	ESTI	ETE	EVA
EVAMO	EVANO	EVATE	EVI	EVO	UTA
UTE	UTI	UTO	ONO	ERE	ERO
ERAI	ERA	EREMO	ERETE	ERANNO	EREI
ERESTI	EREBBE	EREBBERO	EREMMO	ERESTE	AMERAI
METTERAI	MANGERAI	LASCERAI	ISTE	ISTI	ITA
ITA	II	I	IMMO	IRAI	IRANNO
IRE	IREBBE	IREBBERO	IREI	IREMMO	IREMO
IRESTE	IRESTI	IRETE	ITA	ITE	ITI
ITO	IVA	IVAMO	IVANO	IVATE	IVI
IVO	ISSE	ISSERO *	ISSI	ANO	IAMO
IATE	ISCA	ISCANO	ISCE	ISCI	ISCO
ISCONO	CO	GO	CA	GA	GHE
CHE	CIA	RAI	RA	REMO	RETE
RANNO	RESTI	REBBE	REMMO	RESTE	REBBERO
SERO	ETTERO				

German Prefixes

The following German prefixes are processed by SCRIPT/VS during stem processing.

ABER	ABTAST	ABSATZ	AB
ACHTUND	ACHT	AKZEPT	ALLGEMEIN
ALT	ANLAGE	ANLEIHE	ANLERN
ANTI	ANTRAGS	AN	AUFTRAG
AUF	AUSFUHR	AUSSEN	AUS
BAGATELL	BAHN	BANK	BAR
BAUSPAR	BEDARFS	BEI	BEREIT
BESCHLUSS	BESITZ	BESTELL	BETRIEBS
BILANZ	BINNEN	BRIEF	BRUTTO
BUCH	BUDGET	DAR	DA
DEFIZIT	DEPOT	DIENTST	DIREKT
DISKONT	DOLLAR	DOPPEL	DREH
DREIUND	DREI	DRITTEL	DRITT
DRUCK	DURCHFUHR	DURCH	EIGENTUMS
EINFUHR	EINKAUF	EINLAGE	EINNAHME
EINSATZ	EINSPRUCHS	EINUND	EINZEL
EIN	ELEKTRO	ENERGIE	ENTGEGEN
ERSATZ	ERWERBS	EUROPA	EXEKUTIV
EXISTENZ	EXPORT	FABRIK	FACH
FAHRT	FAHR	FEHL	FEIN
FERNMELDE	FERN	FEST	FILIAL
FINANZIELL	FINANZ	FLUGZEUG	FLUG
FLUT	FORT	FRACHT	FREI
FREMD	FRIST	FRUEH	FUENFUND
FUENF	FUERSORGE	FUNKTIONAERS	GANZ
GARANTIE	GAST	GAS	GEBAEUDE
GELD	GEMEINDE	GEMEIN	GENERAL
GEPAECK	GESAMT	GESETZ	GEWAEHR
GEWERBE	GEWERBS	GEWINN	GE
GIRO	GLEICH	GRENZ	GROSS
GRUENDSAETZ	GRUND	GUT	HAFT
HALBJAHRES	HALB	HAND	HAUPT
HAUS	HEIMAT	HEIM	HERAB
HERAN	HERAUF	HERAUSGE	HERAUS
HERBEI	HEREIN	HERNACH	HERUEBER
HERUM	HERUNTER	HERVOR	HER
HILFS	HINAB	HINAN	HINAUF
HINAUS	HINEIN	HINDURCH	HINUEBER
HINUNTER	HINWEG	HINZU	HIN
HOCHSCHUL	HOCH	HOECHST	HONORAR
HUNDERT	IMPORT	INDIVIDUAL	INDUSTRIE
INITIATIV	INNEN	INSOLVENZ	INTERZONEN
INVENTAR	INVESTIV	INVESTMENT	IN
JUGEND	JUSTIZ	KABEL	KAMPF
KAPITAL	KARTELL	KASSE	KAUF

KERN	KLAGE	KLEIN	KLINIK
KLIMA	KOLLEKTIV	KOMMUNAL	KOMPROMISS
KONJUNKTUR	KONKURRENZ	KONKURS	KONTO
KONTROLL	KONZERN	KRAFT	KREDIT
KURIS	KULTUR	KULTUS	KUNSTSTOFF
KUNST	KURS	KURZ	KUR
LANG	LEHR	LIZENZ	LOHN
LOS	LUFT	LUXUS	MACHT
MAGNET	MANAGEMENT	MARKT	MATERIAL
MEHR	MERK	METALL	MIET
MIKRO	MINDEST	MINIMAL	MISS
MITTEL	MIT	MODELL	MONOPOL
MONTAN	NACHFRAGE	NACHSCHUB	NACHWUCHS
NACH	NATIONAL	NETTO	NEUNUND
NEUN	NEU	NICHT	NOMINAL
NORD	NORMAL	NOT	NUTZ
OBEN	OBER	OPTIMAL	ORTS
OST	PACHT	PAKET	PARALLEL
PARTEI	PATENT	PAUSCHAL	PERSONAL
PFAND	PFLICHT	PLAN	PORTO
POST	PREIS	PRESSE	PRIVAT
PROBE	PROBLEM	PROFIT	PROJECT
PROTEST	PROZENT	PROZESS	PUNKT
QUER	RAND	RANG	RATS
REFORM	REGEL	REGIONAL	REIN
REISE	REKLAME	REPARATUR	REPRAESENTATIV
REPORT	RESERVE	REST	RISIKO
ROHSTOFF	ROH	RUECKWAERTS	RUECK
RUHEGELD	RUHE	RUNDFUNK	RUND
SACH	SALDO	SAMMEL	SCHATZ
SCHECK	SCHEIN	SCHLUSS	SCHNELL
SCHREIB	SCHRIFT	SCHUL	SCHUTZ
SECHSUND	SEE	SELBST	SICHT
SIEBENUND	SOFORT	SOLL	SONDER
SOZIAL	SPAR	SPERR	SPEZIAL
SPITAL	SPRACH	STADT	STAHL
STANDARD	STAMM	START	STIMM
STOER	STRAF	STRAHL	STREIK
STROM	STRUKTUR	STUECK	SUBSTANZ
SUED	SUPER	SYSTEM	TARIF
TAT	TAUSCH	TEIL	TELEFON
TELEGRAM	TENDENZ	TERMIN	TEST
TEXTIL	TON	TOTAL	TRANSPORT
TREUHAND	TREUHAENDER	UEBERNAHME	UEBERSCHUSS
UMLAUFS	UMSATZ	UMTAUSCH	UMWELT
UM	UNFALL	UN	UR
VALUTA	VERDIENST	VERGLEICH	VERKAUFS
VERKEHR	VERLADE	VERLUST	VERMAECHTNIS
VERSAND	VERTRIEBS	VIEL	VIERTEL
VIERUND	VIER	VOLL	VORAN
VORAUFG	VORAUFG	VORJAHRES	VORSORGE
VORWAERTS	VOR	WACHSTUMS	WAHL
WANDEL	WECHSEL	WEG	WEHR
WEIHNACHTS	WELT	WERBE	WERT
WEST	WIDER	WOHL	WOHN
ZEHN	ZENTRAL	ZIEL	ZINS
ZIVIL	ZOLL	ZURUECK	ZUSATZ
ZUSCHUSS	ZUWACHS	ZU	ZWECK
ZWEIG	ZWEIUND	ZWEI	

Spanish Prefixes

The following Spanish prefixes are processed by SCRIPT/VS during stem processing.

CONTRA	RECOMP	PRES#	PRES\$	SUBS#
SUBS\$	TRANS	ANTI	COMP	CONP
PRE>	REM#	REM\$	REN#	REN\$
RES#	RES\$	CON	DES	IMB
IMP	INB	INP	IRR	PRE
RE>	SUB	IN	RE	

Chapter 9. Performance Considerations

Chapter 9. Performance Considerations	105
Storage Usage	105
SCRIPT Command Options	105
FPASSES(n) Option	106
INDEX Option	106
SEPMASSTR Option	106
SPELLCHK Option	106
TWOPASS Option	107
Mapped Page Segments	107
Changing the NOMAP Default	107
Removing Extra Languages from the GML Starter Set	107
SCRIPT/VS in the ATMS-III Environment	108
Tuning ATMS-III for SCRIPT/VS	108

Chapter 9. Performance Considerations

Several facilities provided by SCRIPT/VS can increase the system resources used in formatting documents. This chapter contains information on how to improve system performance.

Storage Usage

The following list gives some hints about how to minimize storage used by DCF:

1. Large tables use large amounts of storage. Split large tables into smaller tables, or use the BREAK parameter on the .TA [Table] control word to split the table between rows, thereby returning storage to the system. Refer to the *DCF: SCRIPT/VS Language Reference* for information about the BREAK parameter of the .TA control word.
2. Defining fonts, symbols, and macros uses storage and slows down processing time. Do not define these items if you are not using them.
3. Use *local* symbols in macros whenever possible since they don't go into the symbol table.
4. Check the *DCF: GML Starter Set Implementation Guide* for tips about returning storage at the end of a macro.
5. If using rotated areas, specify the FONT parameter on the .DA [Define Area] control word. This keeps DCF from using storage to create a "dummy" font every time the area is started.
6. Place page segments in the system library if possible rather than including them inline.

Note: If formatting for a PostScript device, graphics take up space because they are always included inline due to restrictions in the PostScript language.

SCRIPT Command Options

These options of the SCRIPT command can significantly increase the CPU resources needed to format a document:

- FPASSES(n) — Process the document with multiple passes
- INDEX — Create an index
- SEPMASTR — Create separate masters from one file without manipulating the output file
- SPELLCHK — Perform spelling verification
- TWOPASS — Process the document twice.

Each function must be explicitly specified; none are defaults. The effect on performance of each option is independent of the other options and is discussed separately.

FPASSES(n) Option

The FPASSES(n) option causes the document to be formatted a specified number of times. This includes all imbedded files and macros. The IBM-supplied maximum number of formatting passes is 4, but the maximum value can be changed at your installation. See Chapter 10, "Tailoring SCRIPT/VS for Your Installation" on page 111. Only the SCRIPT/VS symbol table and table of contents file are saved from each pass; formatted output is produced only on the last pass.

Usually, two formatting passes are enough to correctly resolve cross-references and symbols. When the FPASSES(n) option is used to specify more than two formatting passes, the system resources are increased for each additional formatting run. But, if your table of contents has incorrect page numbers, or cross-references and symbols are not resolved correctly, using FPASSES(n) to specify more than two passes should resolve them correctly.

INDEX Option

When the INDEX option is included, index terms specified in the body of the document are saved and sorted to produce an index in the back of the document. Producing a large index can consume significant amounts of both virtual storage and processor time, because the index entries are kept in storage and are sorted dynamically.

Because an INDEX is often unnecessary for draft copies of a document, the INDEX option can simply be omitted; the index terms specified in the document will be ignored.

SEPMASSTR Option

The SEPMASSTR option allows you to create separate masters from one output file. You can mark these masters to represent different colors or portions of a multi-part form. These masters will then be used by a printing vendor in the offset printing process.

When the SEPMASSTR option is specified, each page that contains data to be included in a separate master must be processed multiple times, thus using significant amounts of processor time.

SPELLCHK Option

When spelling verification is enabled, each occurrence of every word in the document being formatted is checked against the active dictionaries.

Because spelling verification significantly increases the processor time required to format a document, it should be used only occasionally. Often it is sufficient to perform spelling verification only twice: once, when the document is first created, to find entry errors, acronyms, and valid words that are not in the dictionaries, and again, just before the final formatting runs, to catch any errors made while updating or revising the document.

TWOPASS Option

The TWOPASS option causes the document being formatted, including all imbedded files and macros, to be processed twice. Only the SCRIPT/VS symbol table and table of contents file are saved from the first pass; formatted output is produced only on the second pass.

The TWOPASS option must be used when an automatically generated table of contents is placed in the front of a document or when reference is made to symbols that are set later in the document. These are called *forward references*, because the value of the symbol is used before the symbol is defined.

Not unexpectedly, the TWOPASS option roughly doubles the system resources consumed in formatting a document. However, unresolved forward references are often acceptable in early proofing versions of a document. Similarly, the table of contents can often be moved temporarily to the back of the document. In these cases, the TWOPASS option can be omitted for all but the final formatting runs, when the table of contents is correctly placed.

Mapped Page Segments

When printing page segments on a 3820 Page Printer or a 3800 Printing Subsystem Model 3 or 6, use the MAP parameter on the .SI [Segment Include] control word to cause page segments to be downloaded to printer storage at the start of printing that page, unless that page segment already resides in printer storage. If you MAP a page segment and include it multiple times on the same page, it will be downloaded only once to the printer. It is more efficient to use mapped page segments if the same segment is included more than once in a document.

Changing the NOMAP Default

The IBM-supplied default is to not map page segments, but you can change the default by modifying the DSMLPPDT file at installation time. See "DSMLPPDT: Physical Device Tables" on page 136 for information on how to change this file. You can also override the default by specifying MAP on the .SI [Segment Include] control word. Refer to the *DCF: SCRIPT/VS Language Reference* for information on using the .SI control word.

Removing Extra Languages from the GML Starter Set

If your documents only use one language, you can delete the macros for the languages you don't need from the GML starter set maclib.

There are four groups of macros that deal with languages:

Macro Name	Language Function
DSM#I01 through DSM#I18	Input languages
DSM#L01 through DSM#L18	Language dependent
DSM#M01 through DSM#M18	Messages
DSM#S01 through DSM#S18	Symbols

See Table 8 on page 172 for a list of the two-number codes and their associated languages.

To remove the macros in a CMS environment, use the MACLIB command. To remove the macros in an MVS or VSE environment, delete the members.

This should improve system performance because SCRIPT/VS doesn't have to read through unnecessary coding.

Note: If you do delete the macros from the starter set maclib, keep an unmodified starter set available.

SCRIPT/VS in the ATMS-III Environment

In the ATMS-III environment, CICS provides facilities for creating and editing documents, and SCRIPT/VS can be used to format these documents. If you are an ATMS user, you can invoke SCRIPT/VS to format your documents in any one of three ways:

- **Online formatting:** SCRIPT/VS can be invoked at the terminal to format a document currently residing in ATMS working storage. The output is placed in CICS/VS auxiliary temporary storage; it can then be transferred to a printer or reviewed at your terminal.
- **Peripheral queue:** Requests can be placed on a special queue for deferred document processing by SCRIPT/VS. A single CICS task is used to process all such queues.
- **Batch formatting:** A batch job can be submitted to format a document using SCRIPT/VS if the document and all imbedded documents and macros reside within the Document Library Facility.

Tuning ATMS-III for SCRIPT/VS

Five parameters provided with the ATMS-III system-generation macro DOKVA can be used to regulate SCRIPT/VS in the CICS environment. The parameters are:

SPA	Limits the amount of virtual storage used by SCRIPT/VS when formatting documents submitted to a peripheral queue. SPA gives the number of 8K-byte blocks that can be obtained per document, beyond an initial 64K-byte block.
SPP	Limits the number of output pages that can be produced when formatting documents submitted to a peripheral queue. SPP gives the number of pages permitted.
STA	Limits the amount of virtual storage used by SCRIPT/VS when formatting documents online from a terminal. STA gives the number of 8K-byte blocks that can be obtained per document, beyond an initial 64K-byte block.
STO	Limits the number of concurrent online users of SCRIPT/VS.
TSP	Limits the number of output pages that can be produced when formatting documents online from a terminal. TSP gives the number of pages permitted.

Further information on tuning ATMS-III in the CICS environment can be found in the *ATMS-III Program Reference Manual*.

Chapter 10. Tailoring SCRIPT/VS for Your Installation

Chapter 10. Tailoring SCRIPT/VS for Your Installation	111
Procedure for Tailoring SCRIPT/VS	111
Changing the Resident Dictionary	112
Source Files Provided with SCRIPT/VS	112
DSMCSMDM: Message Definitions	113
The DSMMSG Macro	114
The DSMSMSV Macro	115
DSMTSMDM: Additional Message Definitions for TSO	115
The IKJTSMSG Macro	116
DSMCSPDB: Phrase Definitions	116
The DSMPDB Macro	119
DSMeeUOT: Environment-Dependent Options Tables	119
The DSMDEF Macro	119
The DSMSCO Macro	121
Changing Default SCRIPT Command Options	123
The DSMATDEF Macro (ATMS-III Only)	124
The DSMCMDEF Macro (CMS Only)	126
The DSMLLDEF Macro (MVS and VSE Only)	127
The DSMTSDEF Macro (TSO Only)	127
DSMLPBCS: Box Character Sets	130
The DSMBCS Macro	130
The DSMHHCC Macro	131
DSMLPLDT: Logical Device Definitions	131
The DSMLDT Macro	131
The DSMSLDT Macro	136
DSMLPPDT: Physical Device Tables	136
DSMPDT Macro	136
DSMSPDT Macro	140
DSMTFFIB: Line Printer Font Information	140
DSMFIB Macro	141
Line Device Width Tables	143
DSMTRNLT: National Language Character Definitions	144
DSMACTT Macro	144
DSMWDPT Macro	146
DSMHCC Macro	147
Index Sort Sequence Macros	147
DSMIXTT2 Macro	147
DSMIXTT1 Macro	149
Replacing the IBM-Supplied Dictionaries	150
Required Fonts for Page Printers	151
Required Fonts for PostScript Devices	153

Chapter 10. Tailoring SCRIPT/VS for Your Installation

SCRIPT/VS can be modified to fit more closely the needs of your installation. DCF provides assembler language source files for several SCRIPT/VS modules; these files contain table specifications that determine default values, message text, logical-device characteristics, and so on.

Note: Central Programming Service support and maintenance are available for SCRIPT/VS. However, support and maintenance *are not* available if the user makes any modifications to SCRIPT/VS that are not in accordance with the procedures listed in this section.

Procedure for Tailoring SCRIPT/VS

To modify SCRIPT/VS, you and your systems programmer must perform the following steps:

1. Decide what changes to SCRIPT/VS you want to make for your installation.
2. Modify the appropriate table specifications in the IBM-supplied source files.

The MVS System Modification Program/Extended (SMP/E) and the VSE Maintain System History Program (MSHP) provide an automatic update process when you modify macros or source modules.

When running in a non-XA CMS environment or with a CMS level of 5 or earlier, you must issue the following command before you reassemble a modified source file:

```
GLOBAL MACLIB DSMMAC4 CMSLIB
```

When running in an XA CMS environment or with a CMS level greater than 5, you must issue the following command before you reassemble a modified source file:

```
GLOBAL MACLIB DSMMAC4 DMSGPI
```

3. Reassemble the modified source files with the IBM-supplied macro library.

When changing any files supplied in source form, you must delete the following lines of text from the file if you plan to assemble the file with the F assembler.

```
DSMxxxxx AMODE 31          Assign addressing mode
DSMxxxxx RMODE ANY        Assign residency mode
```

where xxxxx are the 5 characters used to describe the file (for example, DSMLPLDT for Logical Device Tables). You do not need to delete those lines if you are running on an XA or ESA system with the H assembler.

If you want to run DCF in XA exploitation mode, you must use the H Assembler to assemble these modules.

If you don't have the H Assembler, you can still assemble these modules with the F Assembler, but you must delete the A mode, R mode compiler options. The resulting code will not run in an XA environment.

4. Linkedit your SCRIPT/VS load module again using the object files created by the assembler and the IBM-supplied object library.
5. Run the DSMGND40 and DSMGNS40 EXECS.

6. Install a new load module in the appropriate library.
7. Inform all users at your installation of the changes you have made.

Changing the Resident Dictionary

TSO: If you use DCF in the TSO environment, you can improve performance by only including the languages you intend to use in the TSO load module. English American is included in the TSO load module by default. To change the resident dictionary or to add more dictionaries to the load module:

1. Edit the file named SCRIPT.R40.DCFSAMP(DSMTSDCT).
2. Delete all languages you don't want included in the load module.
3. Run that job.

CMS: If you use DCF in the CMS environment, you choose your resident dictionary at installation time with the I5748XX9 EXEC. You will be prompted to include one or more dictionaries in your DCF system by entering a 3- or 4-character code. If you do not want to include any dictionaries, you just press ENTER.

Source Files Provided with SCRIPT/VS

The following assembler language files are provided in source form:

DSMCSDM (Message Definitions). The tables in this file give the text of messages issued by SCRIPT/VS.

DSMTSDM (Message Definitions). The tables in this file give the text of additional messages issued by SCRIPT/VS in the TSO environment.

DSMCSPDB (Phrase Definitions). The tables in this file give the text of literal phrases used by SCRIPT/VS in formatting documents.

DSMeeUOT (Environment-dependent Options). The tables in this file define the names of the SCRIPT command options and establish default values for those options. There is a separate file for each operating system environment under which SCRIPT/VS operates:

DSMATUOT	ATMS-III with CICS/VS
DSMCMUOT	CMS with VM/370
DSMVXUOT	CMS with VM/XA or VM/ESA
DSMLDUOT	DLF with VSE
DSMLOUOT	DLF with MVS
DSMTSUOT	TSO with MVS

DSMLPBCS (Box Character Sets). The tables in this file define the box character sets that can be specified with the CHARS parameter of the .BX [Box] control word, and establish the characters that will be used to construct boxes and rules on line devices.

DSMLPLDT (Logical Device Tables). The tables in this file define the logical devices that can be specified with the DEVICE option of the SCRIPT command, and establish the characteristics of those logical devices.

DSMLPPDT (Physical Device Tables). The tables in this file define the physical devices that are referenced by the logical devices, and establish the characteristics of those physical devices.

DSMTFFIB (Line Printer Font Information). The tables in this file define the fonts that can be specified with the CHARS option of the SCRIPT command for line devices, and establish the characteristics of those fonts.

DSMTRNLT (National Language Character Tables). The tables in this file define the correspondence between the internal and external representations of accented characters, determine which characters will be treated as word delimiters or punctuation for purposes of spelling verification and hyphenation, and define the index sort sequences for the IBM-supplied languages.

A macro library is also provided for use in assembling these source files. It contains the following macros:

```
DSMACTT
DSMATDEF (CICS only)
DSMBCS
DSMCMDEF (CMS only)
DSMDEF
DSMECOB
DSMEICB
DSMFIB
DSMHCC
DSMIXTT1
DSMIXTT2
DSMLDT
DSMLLDEF (DLF only)
DSMMSG
DSMPDB
DSMPDT
DSMSCO
DSMSLDT
DSMSMSV
DSMSPDT
DSMTSDEF (TSO only)
DSMWDPT
```

DSMCSMDM: Message Definitions

DCF issues a variety of messages during its processing. The file DSMCSMDM defines the messages issued by SCRIPT/VS during the processing of documents.

SCRIPT/VS messages are prefixed with the identifier DSM and are of the form:

DSMmmmmnnnc

mmm	The name of the module that issued the message
nnn	The message number
c	A message-severity code.

The DSM prefix is printed only when the MESSAGE(ID) option of the SCRIPT command is specified.

The message numbers and severity codes, along with the text of the messages, are defined in the file DSMCSMDM.

The DSMMSG Macro

The file DSMCSMDM contains one DSMMSG macro for each SCRIPT/VS message.

The DSMMSG macro is coded as follows:

DSMMSG number,
code,
text,
action

number A three-digit message number.

code A one-character message-severity code. The SCRIPT/VS return code is determined by the highest-severity message issued during processing. The valid **code** values are:

R Reply required (RC=0)
I Informational (RC=0)
W Warning (RC=4)
E Error (RC=8)
S Severe error (RC=12)
T Terminal error (RC=16)

Error messages will terminate processing if the CONTINUE option of the SCRIPT command has *not* been specified. Severe and Terminal messages always terminate formatting.

text Text of the message, including predefined and variable information fields. **text** is coded as a series of quoted text strings and keywords separated by commas and enclosed in parentheses. Keywords cause predefined or variable information to be included in the message. For example:

('DISK ',VAR1,' NOT ACCESSED')

gives a message text including two quoted strings and a variable string provided by the module issuing the message.

The valid keywords are:

ERRBF The contents of ERRBF
ERRBF3 The first three characters of ERRBF
FILNAM The name of the current input file
KPTYPE The current type of keep or float
PAGENO The current page number
STATFIL The name of the new input file
VARn A variable provided by the module issuing the message. **n** is a number between 1 and 5.

VERSION The version of the SCRIPT module.

The maximum length of a quoted text string is 200 characters. The maximum length of the message text, including variable information, is 250 characters.

action Action to be performed when the message is issued. **action** is coded as a series of keywords separated by commas and enclosed in parentheses. For example:

(INPTLINE,PERIOD)

prints the last input line read, along with the message text, and causes a period to be appended to the message text.

The valid keywords are:

CONDID	Print message identifier only if MESSAGE(ID) is specified
IMBEDTRC	Print a trace of imbedded files after the message
INPTLINE	Print the last input line read after the message
PERIOD	Append a period to the message text
UNCONDID	Print message identifier regardless of MESSAGE(ID) option.

For example, message 317 provided with SCRIPT/VS is coded as:

```
DSMMSG 317,S,('DISK ',VAR1,
  ' IS READ-ONLY OR ',
  'NOT ACCESSED'),
(CONDID,PERIOD)
```

This is a severe error message, coded as "S"; it will terminate SCRIPT/VS processing when issued. The message contains one variable, coded "VAR1," which the module issuing the message will replace with a value.

The DSMSMSV Macro

The file DSMCSMDM contains one DSMSMSV macro following the last DSMMSG macro specification.

The DSMSMSV macro is coded as follows:

DSMSMSV

This macro creates a table of defined message numbers. DSMSMSV must be the last macro coded in DSMCSMDM and requires no parameters.

DSMTSMDM: Additional Message Definitions for TSO

DCF issues a variety of messages during its processing. The file DSMTSMDM defines additional messages issued by SCRIPT/VS during the processing of documents in the TSO environment.

The IKJTSMMSG Macro

The file DSMTSMDM contains one IKJTSMMSG macro for each TSO message issued by SCRIPT/VS in the TSO environment.

The IKJTSMMSG macro is coded as follows:

```
IKJTSMMSG text,  
          number  
          [,chain]
```

text Text of the message, including the message identifier and variable information fields. **text** is coded as a series of quoted text strings separated by commas and enclosed in parentheses. Null strings are used to represent variable information. For example:

```
( 'DSMTS0358S LOCATE RC=', )
```

gives the message identifier and text as a quoted string, followed by a null string to represent a variable provided by the module issuing the message.

number Message number, prefixed by an "S." For example:

```
S358
```

chain Number of another message to which this message is chained. The other message must contain a plus sign (+) as the last text character.

For example, message 357 provided with SCRIPT/VS is coded as:

```
IKJTSMMSG ( 'DSMTS0357S DATASET ',,  
            ' NOT FOUND IN CATALOG+',  
            S357
```

The null string (coded as ",") is replaced with a data set name when the message is printed.

Message 358 is chained to message 357 (as indicated by the suffix plus sign) and is coded as:

```
IKJTSMMSG ( 'DSMTS0358S ',  
            ' 'LOCATE' ' RC = ', ),  
            S358,S357
```

The null string, coded as ",)" is replaced with a return code when the message is printed.

TSO prints message 358 if the user responds to message 357 with a question mark (?).

DSMCSPDB: Phrase Definitions

SCRIPT/VS uses a variety of literal phrases during its processing. Some of these phrases are used as variable information in messages, and some are inserted into composed documents as output text.

The following phrases have been defined:

- 1** **SYMBOL**
 Inserted into message 574
- 2** **MACRO**
 Inserted into message 574
- 3** **TRANSLATION TABLE**
 Inserted into messages 574 and 593
- 4** **STACK**
 Inserted into messages 574 and 593
- 5** **ENVIRONMENT**
 Inserted into messages 574 and 593
- 6** **RULE**
 Inserted into messages 574 and 593
- 7** **AREA**
 Inserted into messages 574 and 593
- 8** **CONTENTS**
 The default heading for the table of contents generated with the .TC [Table of Contents] control word
- 9** **INDEX**
 The default heading for the index generated with the .IX [Index] control word
- 10** **PAGE**
 Used in the default running title, if present
- 11** **See**
 Used to prefix index references if no nonreference index terms appear with the index reference
- 12** **See also**
 Used to prefix index references if other nonreference index terms appear with the index reference
- 13** **CONTROL TRACE**
 Used by the .IT [Input Trace] control word when STEP has been specified
- 14** **TERMINAL INPUT**
 Used by the .IT [Input Trace] control word when STEP has been specified
- 15** **TOP**
 Inserted into message 708
- 16** **BOTTOM**
 Inserted into message 708
- 17** **LEFT**
 Inserted into message 708
- 18** **RIGHT**
 Inserted into message 708
- 19** **(UNKNOWN)**
 Used for a *named* area whose original *name* has been redefined
- 20** **LINE**
 Inserted into message 482

- 21** LINES
Inserted into message 482
- 22** DEFINITION
Inserted into message 531
- 23** PROCESSING
Inserted into message 531
- 24** -
Index Page Range Separator
- 25** ,
Index Page List Separator⁵
- 26** TABLE
Inserted into messages 441, 444, 528, 574, 593, and 595
- 27** ROW
Inserted into messages 441, 444, 574, 593, 595, and 716
- 28** HEADER
Inserted into message 450
- 29** FOOTER
Inserted into message 450
- 30** FLOAT
Inserted into message 528
- 31** KEEP
Inserted into message 528
- 32** FOOTNOTE
Inserted into message 518
- 33** TABLENAME
Inserted into message 598
- 34** ROWNAME
Inserted into message 598
- 35** CELL NUMBER
Inserted into message 598
- 36** ROWS
Inserted into message 716
- 37** WIDTH
Inserted into message 385
- 38** WEIGHT
Inserted into message 385
- 39** SHADE
Inserted into messages 574, 593, and 595

⁵ The index page list separator is a comma followed by a blank.

The DSMPDB Macro

The file DSMCSPDB contains one DSMPDB macro for each SCRIPT/VS phrase.

The DSMPDB macro is coded as follows:

```
DSMPDB NO=number,  
      TXT='phrase'
```

number Phrase number.

phrase Literal phrase.

For example, the DSMPDB specification for phrase number 8 provided with SCRIPT/VS is coded as:

```
DSMPDB NO=8,  
      TXT='CONTENTS'
```

This phrase is used as the default level-one heading for the first page created by the .TC [Table of Contents] control word.

If your users find this phrase too terse, you might want to change its definition:

```
DSMPDB NO=8,  
      TXT='Table of Contents'
```

This will change the default heading for the .TC control word.

DSMeeUOT: Environment-Dependent Options Tables

During SCRIPT/VS processing, a variety of initial and default values are required in each operating environment. These values are located in one of the following files, which correspond to the operating system in which SCRIPT/VS is operating:

DSMATUOT	ATMS-III with CICS/VS
DSMCMUOT	CMS with VM
DSMVXUOT	CMS with VM/XA or VM/ESA
DSMLDUOT	DLF with VSE
DSMLOUOT	DLF with MVS
DSMTSUOT	TSO with MVS.

The DSMDEF Macro

The DSMDEF macro is used in the DSMeeUOT files to specify several environment-independent constants. These constants are:

- The name of the host operating system. This is used as the value of the SCRIPT/VS system symbol &\$\$SYS..
- The default language for spelling verification, hyphenation, and index sort sequence.
- The initial work space size. This amount of read/write storage is obtained during SCRIPT/VS initialization for input/output buffers, page buffers, symbol and macro definitions, and so on. If this initial allocation is exceeded, additional storage is obtained as needed.

- Default logical device names for output destinations. The SCRIPT output destination options are PRINT, FILE, TERM and CTF. If a logical device name is not also given when a destination option is specified, the logical device named with the DSMDEF macro is used.

Note: TERM cannot be redefined.

- The number of formatting passes. System symbol &\$PSNO contains the number of formatting passes on FPASSES(n).

The DSMDEF macro is coded as follows:

```
DSMDEF SYS=sys-code,
        LANG=lang-code,
        INITSTR=size,
        PRINTLD='ld-name',
        FILELD='ld-name',
        CTFLD='ld-name',
        MAXPASS=num,
        ENGALGO=algorithm
```

sys-code Indicates the host operating system with which SCRIPT/VS will operate. The valid values are:

CICS	For DSMATUOT
CMS	For DSMCMUOT
DOS	For DSMLDUOT
TSO	For DSMTSUOT
VS2	For DSMLOUOT

lang-code Indicates the default language for spelling verification, hyphenation, and index sort sequence. The valid values are:

DAN	Danish
DUTH	Dutch
EAM	American English
ECAN	Canadian English
EUK	United Kingdom English
FCAN	Canadian French
FIN	Finnish
FNAT	French National
GERM	German
ICE	Icelandic
ITAL	Italian
NOR	Norwegian
POR	Portuguese
SPAN	Spanish
SWE	Swedish

The default value is EAM.

size Specifies the size of the initial storage area obtained by SCRIPT/VS for read/write work space. **size** can be given as a number of bytes, a number of K bytes (K equals 1024), or a number of M bytes (M equals 1,048,576). Therefore, the following specifications are all valid:

```
SIZE=25000 (25,000 bytes)
SIZE=75K   (76,800 bytes)
SIZE=2M    (2,097,152 bytes)
```

The default value is 64K.

ld-name Identifies a logical device name to be used when the options entered with the SCRIPT command specify only a destination. The four destinations are PRINT, FILE, TERM, and CTF. The values given with the PRINTLD, FILELD, TERMLD, and CTFLD keywords must be defined as logical devices with the DSMLDT macro in the DSMLPLDT module. The default values are:

```
PRINTLD 1403W6
FILELD  1403W6
CTFLD   STAIRS
```

num Specifies the maximum number of passes to allow when formatting a document. The valid range is from 2 to 255. If the value is a character or a number less than 2 or greater than 255, a compilation error occurs. The default value is 4.

algorithm Specifies which algorithmic hyphenator SCRIPT/VS should use for the three English languages: English American, English Canadian, and English United Kingdom. This value can be:

```
old      This is the default. Specifies that SCRIPT/VS should use
           the algorithmic hyphenator available before Release 4.0.

new      Specifies that SCRIPT/VS should use the algorithmic
           hyphenator available with Release 4.0.
```

The DSMSCO Macro

The DSMSCO macro is used in the DSMeeUOT modules to give the names of all SCRIPT command options, plus suboptions of the MESSAGE, PAGE, and PSOUT options.

The DSMSCO macro is coded as follows:

```
DSMSCO [BIND='name']
[,CHARS='name']
[,CONT='name']
[,CTF='name']
[,DDUT='name']
[,DEST='name']
[,DEVICE='name']
[,FILE='name']
[,FONTLIB='name']
[,FPASSES='name']
[,INDEX='name']
[,LIB='name']
[,MESSAGE='name']
[,ID='name']
[,DELAY='name']
[,TRACE='name']
[,NOCONT='name']
[,NODDUT='name']
[,NOPROF='name']
[,NOSPIE='name']
[,NOSEGLB='name']
[,NOSORCD='name']
[,NOWAIT='name']
[,NUMBER='name']
[,OPTIONS='name']
[,PAGE='name']
[,FROM='name']
[,PROMPT='name']
[,TO='name']
[,FOR='name']
[,ONLY='name']
[,PRINT='name']
[,PROFILE='name']
[,PSOUT='name']
[,ASCII='name']
[,EBCDIC='name']
[,QUIET='name']
[,SEARCH='name']
[,SEGLIB='name']
[,SEPM='name']
[,SORCDD='name']
[,SPELL='name']
[,STOP='name']
[,SYOFF='name']
[,SYON='name']
[,SYSVAR='name']
[,TERM='name']
[,TLIB='name']
[,TWOPASS='name']
[,UNIFORM='name']
[,UPCASE='name']
```

name The name by which the option or suboption will be known on the SCRIPT command. If a keyword is not specified, the corresponding SCRIPT command option will not be recognized for this environment.

Notes:

1. The NOSORCDD and SORCDD options are only supported in the TSO environment.
2. The DEST, LIB, NOSPIE, OPTIONS, TERM, and TLIB options are not supported in the DLF environments.

Changing Default SCRIPT Command Options

The DSMESDCO entry point in the DSMeeUOT modules is used to establish default SCRIPT command options. The options specified at this entry point are processed before any options entered on the SCRIPT command line. Any options entered on the SCRIPT command line override any default options specified at this entry point.

In CMS (DSMCMUOT) and ATMS (DSMATUOT), default SCRIPT options specified at the DSMESDCO label must be tokenized in the following manner:

- Each word must be left-justified in an 8-byte field and padded with blanks on the right.
- A parenthesis must be coded in a separate 8-byte field.
- The last word must be followed by a terminator—8 bytes of hexadecimal FF.

As an example, if you want to specify “BIND(25MM 40MM) PROFILE(DSMPROF4)” as default options in CMS and ATMS, the following defined constants should be inserted at the DSMESDCO entry point:

```
DSMESDCO EQU      *
                  DC      CL8'BIND'
                  DC      CL8'('
                  DC      CL8'25MM'
                  DC      CL8'40MM'
                  DC      CL8')'
                  DC      CL8'PROFILE'
                  DC      CL8'('
                  DC      CL8'DSMPROF4'
                  DC      CL8')'
                  DC      8X'FF'
```

In TSO (DSMTSUOT), default SCRIPT options specified at the DSMESDCO entry point must be specified in the following manner:

- Options are specified as a character string with at least one blank between each option.
- Any single quotation marks must be doubled.
- The character string must be followed by a terminator—1 byte of hexadecimal FF.

As an example, if you want to specify “BIND(25MM 40MM) PROFILE('USERID.DSMPROF4.TEXT')” as default options in TSO, the following defined constants should be inserted at the DSMESDCO entry point:

```

DSMESDCO EQU      *
                DC   C'BIND(25MM 40MM)'
                DC   C'PROFILE ('USERID.DSMPROF4.TEXT')
                DC   X'FF'

```

In DLF (DSMLDUOT and DSMLUOT), default SCRIPT options specified at the DSMESDCO entry point must be specified in the following manner:

- Each word must be left-justified in an 8-byte field and padded with blanks on the right.
- A parenthesis must be coded in a separate 8-byte field.
- Periods in data set names must also be coded in separate 8-byte fields.
- Any single quotation marks must be doubled and coded in separate 8-byte fields.
- The last word must be followed by a terminator—8 bytes of hexadecimal FF.

As an example, if you want to specify "BIND(25MM 40MM) FILE(DSN('USERID.LIST'))" as default options in DLF in the MVS operating environment, the following defined constants should be inserted after the DSMESDCO label:

```

DSMESDCO EQU      *
                DC   CL8'BIND'
                DC   CL8'('
                DC   CL8'25MM'
                DC   CL8'40MM'
                DC   CL8')'
                DC   CL8'FILE'
                DC   CL8'('
                DC   CL8'DSN'
                DC   CL8'('
                DC   CL8'    '
                DC   CL8'USERID'
                DC   CL8')'
                DC   CL8'LIST'
                DC   CL8'    '
                DC   CL8')'
                DC   CL8')'
                DC   8X'FF'

```

The DSMATDEF Macro (ATMS-III Only)

The DSMATDEF macro is used in the DSMATUOT module to specify several default values for the ATMS-III environment. These defaults are:

- The name of the default macro library. This is the name of a partitioned data set (MVS) or phase (VSE) that contains SCRIPT/VS macros. This library will be used if the LIB option of the SCRIPT command is not specified.
- The name of a partitioned data set that contains four objects. This data set name will be used if the FONTLIB option of the SCRIPT command is not specified.
- The name of the default page segment library. This is the name of a partitioned data set that will be used if the SEGLIB option of the SCRIPT command is not specified.

The DSMATDEF macro is coded as:

```
DSMATDEF MACLIB='op-num',  
          FNLIB42='font-name',  
          PSLIB42='seg-name',  
          FNLIB40='font-name',  
          PSLIB40='seg-name',  
          FNLIB38='font-name',  
          PSLIB38='seg-name',  
          FNLIB20='font-name',  
          PSLIB20='seg-name',  
          FNLIBPS='font-name'
```

op-num The operator number to be used if the LIB option of the SCRIPT command is not specified. The value must be enclosed in quotation marks.

font-name Name of a partitioned data set (only the first-level qualifier) or VSE core image library to be used if the FONTLIB option of the SCRIPT command is not specified. The value must be enclosed in quotation marks.

The FNLIB42 keyword gives the default name for the 4250 printer physical device; the default value is FONT4250.

The FNLIB40 keyword gives the default name for the IBM LaserPrinter 4028 physical device; the default value is FONT300.

The FNLIB38 keyword gives the default name for the 3800 Printing Subsystem Model 3 physical device; the default value is FONT38PP.

The FNLIB20 keyword gives the default name for the 3820 Page Printer physical device; the default value is FONT3820.

The FNLIBPS keyword gives the default name for the PostScript physical device; the default is FONTPS.

Note: ATMS does not support PostScript output from DCF.

seg-name Name of a partitioned data set (only the first-level qualifier) or VSE core image library to be used if the SEGLIB option of the SCRIPT command is not specified. The value must be enclosed in quotation marks.

The PSLIB42 keyword gives the default name for the 4250 printer physical device; the default value is NOSEGLIB.

The PSLIB40 keyword gives the default name for the IBM LaserPrinter 4028 physical device; the default value is NOSEGLIB.

The PSLIB38 keyword gives the default name for the 3800 Printing Subsystem Model 3 physical device; the default value is NOSEGLIB.

The PSLIB20 keyword gives the default name for the 3820 Page Printer physical device; the default value is NOSEGLIB.

The DSMCMDEF Macro (CMS Only)

The DSMCMDEF macro is used in the DSMCMUOT module to specify several default values for the CMS environment. These defaults are:

- The name of the default macro library. This is the file name of a CMS maclib file to be used if the LIB option of the SCRIPT command is not specified.
- The name of the default object library. This is the file name of a CMS txtlib file to be used if the TLIB option of the SCRIPT command is not specified.
- The type of the default font library. This is the file type of a CMS font library to be used if the FONTLIB option of the SCRIPT command is not specified.
- The type of the segment library. This is the file type of a CMS segment library to be used if the SEGLIB option of the SCRIPT command is not specified.

The DSMCMDEF macro is specified as follows:

```
DSMCMDEF MACLIB='mac-name',  
          TXTLIB='txt-name',  
          FNLIB42='font-name',  
          PSLIB42='seg-name',  
          FNLIB40='font-name',  
          PSLIB40='seg-name',  
          FNLIB38='font-name',  
          PSLIB38='seg-name',  
          FNLIB20='font-name',  
          PSLIB20='seg-name',  
          FNLIBPS='font-name'
```

mac-name File name of a CMS macro library that contains SCRIPT/VS macros. This library will be used if the LIB option of the SCRIPT command is not specified. The value must be enclosed in quotation marks, and the default value is DSMGML4.

txt-name File name of a CMS text library that contains SCRIPT/VS language dictionaries. This library will be used if the TLIB option of the SCRIPT command is not specified. The value must be enclosed in quotation marks, and the default value is DSMHYLIB.

font-name File type of a set of CMS files that contains fonts. This file type and a file mode of "*" will be used if the FONTLIB option of the SCRIPT command is not specified.

The FNLIB42 keyword gives the default file type for the 4250 printer physical device; the default value is FONT4250.

The FNLIB40 keyword gives the default file type for the IBM LaserPrinter 4028 physical device; the default value is FONT300.

The FNLIB38 keyword gives the default file type for the 3800 Printing Subsystem Model 3 physical device; the default value is FONT38PP.

The FNLIB20 keyword gives the default file type for the 3820 Page Printer physical device; the default value is FONT3820.

The FNLIBPS keyword gives the default file type for the PostScript physical device; the default value is FONTPS.

seg-name File type of a set of CMS files that contains page segments. This file type and a file mode of “*” will be used if the SEGLIB option of the SCRIPT command is not specified.

The PSLIB42 keyword gives the default file type for the 4250 printer physical device; the default value is PSEG4250.

The PSLIB40 keyword gives the default file type for the 4028 physical device; the default value is PSEG3820.

The PSLIB38 keyword gives the default file type for the 3800 Printing Subsystem Model 3 physical device; the default value is PSEG38PP.

The PSLIB20 keyword gives the default file type for the 3820 Page Printer physical device; the default value is PSEG3820.

The DSMLLDEF Macro (MVS and VSE Only)

The DSMLLDEF macro is used in the DSMLDUOT and DSMLUOT files to specify the library number of the default macro library. This library is used when macros are not found in the user's private or project libraries.

The DSMLLDEF macro is specified as follows:

DSMLLDEF MACLIB = mac-num

mac-num The number of a DLF library that contains SCRIPT/VS macros. The default value is 1314151.

The DSMTSDEF Macro (TSO Only)

The DSMTSDEF macro is used in the DSMTSUOT module to specify several default values for the TSO environment. These default values are:

- The default SYSOUT class for printed output, and the default SYSOUT class for printed output when the UPCASE option of the SCRIPT command is specified.
- The name of the default macro library. This is a fully qualified data set name to be used if the LIB option of the SCRIPT command is not specified.
- The name of the default font library. This is a fully qualified data set name to be used if the FONTLIB option of the SCRIPT command is not specified.
- The name of the default segment library. This is a fully qualified data set name to be used if the SEGLIB option of the SCRIPT command is not specified.
- The space parameters for dynamically allocated data sets. These parameters are similar to the space parameter of the DD JCL statement.
- Dynamic allocation flags. These determine where dynamically allocated data sets are located, whether they will be automatically deallocated, and how unused space will be disposed of.

The DSMTSDEF macro is coded as follows:

```
DSMTSDEF SYSOUT=class,  
    USYSOUT=class,  
    MACLIB=mac-name,  
    FNLIB42=font-name,  
    PSLIB42=segment-name,  
    FNLIB40=font-name,  
    PSLIB40=segment-name,  
    FNLIB38=font-name,  
    PSLIB38=segment-name,  
    FNLIB20=font-name,  
    PSLIB20=segment-name,  
    SPACE=(space),  
    BLKSIZE=size,  
    SAMEVOL=YES|NO,  
    DEALLOC=YES|NO,  
    RELEASE=YES|NO,  
    FNLIBPS=font-name
```

class Default SYSOUT class for use with the PRINT option of the SCRIPT command. The SYSOUT class given with the SYSOUT keyword is used if the UPCASE option of the SCRIPT command is not specified; the SYSOUT class given with the USYSOUT keyword is used if the UPCASE option is specified. The default values are T and A, respectively.

mac-name Name of a macro library that contains SCRIPT/VS macros. This library is used if the LIB option of the SCRIPT command is not specified. The value must be enclosed in quotation marks, and the default value is SCRIPT.R40.MACLIB.

font-name Name of a partitioned data set that contains font objects. This library is used if the FONTLIB option of the SCRIPT command is not specified.

The FNLIB42 keyword gives the default data set name for the 4250 printer physical device; the default value is 'SYS1.FONT4250'

The FNLIB40 keyword gives the default data set name for the IBM LaserPrinter 4028 physical device; the default value is 'SYS1.FONT300'.

The FNLIB38 keyword gives the default data set name for the 3800 Printing Subsystem Model 3 physical device; the default value is 'SYS1.FONT38PP'.

The FNLIB20 keyword gives the default data set name for the 3820 Page Printer physical device; the default value is 'SYS1.FONT3820'.

The FNLIBPS keyword gives the default data set name for the PostScript physical device; the default value is 'SCRIPT.R40.FONTPS'.

segment-name

Name of a partitioned data set that contains segment objects. This library is used if the SEGLIB option of the SCRIPT command is not specified.

The PSLIB42 keyword gives the default data set name for the 4250 printer physical device; the default value is 'SYS1.PSEG4250'.

The PSLIB40 keyword gives the default data set name for the IBM LaserPrinter 4028 physical device; the default value is 'SYS1.PSEG3820'.

The PSLIB38 keyword gives the default data set name for the 3800 Printing Subsystem Model 3 physical device; the default value is 'SYS1.PSEG38PP'.

The PSLIB20 keyword gives the default data set name for 3820 Page Printer physical device; the default value is 'SYS1.PSEG3820'.

space

Space allocation parameters for data sets that are dynamically allocated by SCRIPT/VS. The space parameter is coded in the same manner as the space parameter of the DD JCL statement:

```
SPACE=(TRK|CYL|BLKS,
        primary,
        secondary,
        directory)
```

TRK|CYL|BLKS

Indicates how space should be allocated. TRK indicates that space should be allocated in tracks. CYL indicates that space should be allocated in cylinders. BLKS indicates that space should be allocated in blocks.

primary

Number of tracks, cylinders, or blocks initially allocated as the primary extent.

secondary

Number of tracks, cylinders, or blocks allocated as the secondary extent if the primary extent is filled.

directory

Number of blocks allocated for the partitioned data set directory.

The default value is (CYL,1,1,1).

size

The BLKSIZE that will be used when dynamically allocating UTILITY type data sets, such as DSMUTMSG, DSMUTTOC, and DSMUTWTF. This BLKSIZE will also be used for allocating the output data set (FILE option) when formatting for line printers or terminals. The value specified must be at least 264 bytes. The BLKSIZE value is not used for the output data set when formatting for page printers and is never used for the DSMITCTF utility data set.

YES|NO

Determines the following characteristics of dynamically allocated data sets:

- **SAMEVOL:** If YES is specified, dynamic data sets will be allocated on the same volume as the primary input file specified on the SCRIPT command. The default value is NO.
- **DEALLOC:** If YES is specified, dynamic data sets will be deallocated when they are closed. The default value is YES.
- **RELEASE:** If YES is specified, unused space in dynamic data sets will be released when the files are closed. The default value is NO.

DSMLPBCS: Box Character Sets

SCRIPT/VS provides several facilities for drawing horizontal and vertical rules. On line devices, rules are constructed from individual characters that together form the best possible representation of a rule. The collection of characters used is called a box character set, and it consists of 11 characters:

```

┌───┴───┐
│       │
├───┬───┤
│       │
└───┬───┘

```

The DSMBCS Macro

The file DSMLPBCS contains one DSMBCS macro for each box character set.

The DSMBCS macro is coded as follows:

```

DSMBCS NAME=name,
      CROSS=code,
      VBAR=code,
      HBAR=code
      [,ULCORN=code]
      [,URCORN=code]
      [,LLCORN=code]
      [,LRCORN=code]
      [,TOPTEE=code]
      [,BOTTEE=code]
      [,LEFTEE=code]
      [,RIGHTEE=code]

```

name Name of the box character set. This name can be used with:

- The DSMFIB macro in the file DSMTFFIB
- The CHAR parameter of the .BX [Box] control word
- The BOX parameter of the .DF [Define Font] control word.

code Character code or codes to be printed for a box character. Either one or two characters can be specified, and they can be specified either as quoted strings, such as '?', or as hexadecimal digits, such as 6F.

For example, the following specifications are all valid:

```
CROSS='+ '  
CROSS=4E  
CROSS='|- '  
CROSS=4F60
```

The CROSS, HBAR, and VBAR keywords are required. The four corner intersection and the four “tee” intersection keywords are optional and default to the code specified for CROSS if not specified.

For example, the box character set used for terminal devices is coded as:

```
DSMBCS NAME=TRM,  
VBAR='| ', HBAR='-', CROSS='+ '
```

Because none of the corner or intersection keywords is specified, the character given with the CROSS parameter is used for all of them.

The box character set used for 1403 printer devices assumes the use of the TN print train, and is coded as:

```
DSMBCS NAME=TNC,  
VBAR=4F, HBAR=BF, CROSS=BF4F,  
ULCORN=AC, URCORN=BC,  
LLCORN=AB, LRCORN=BB,  
LEFTEE=AC4F, RIGHTEE=BC4F,  
TOPTEE=BFBC, BOTTEE=BFAB
```

All 11 box characters are specified. Note that the CROSS and “tee” intersection characters are created by overstriking two other characters on the 1403.

The DSMHHCC Macro

The DSMBCS macro uses the DSMHHCC macro. DSMHHCC is not coded in DSMLPBCS.

DSMLPLDT: Logical Device Definitions

SCRIPT/VS formatting is based on the specific characteristics of a specified (or implied) logical output device. SCRIPT/VS takes into account the characteristics of the physical output device, as well as changeable characteristics such as line spacing and form size. The combination of these fixed (physical) characteristics and dynamic characteristics is called the logical output device, which corresponds to the setup of a particular physical output device. For information about the physical device tables, see “DSMLPPDT: Physical Device Tables” on page 136.

The DSMLDT Macro

The file DSMLPLDT contains one logical device table for each logical output device, specified with a DSMLDT macro. The *DCF: SCRIPT/VS Language Reference* lists the logical devices defined in the IBM-supplied source file. You can change the characteristics of these logical devices by modifying the DSMLDT macro specifications in DSMLPLDT. You can create new logical devices by adding a DSMLDT macro specification to DSMLPLDT.

The DSMLDT macro is coded as follows:

```
[label] LDT LD=logical-device name,  
        PD=physical-device name  
        [,DF=font-name]  
        [,LPI=lpi]  
        [,PITCH=pitch]  
        [,ROTATE=rotation]  
        [,PL=page-length]  
        [,PW=page-width]  
        [,LL=line-length]  
        [,PM=page-margin]  
        [,TM=top-margin]  
        [,BM=bot-margin]  
        [,HM=head-margin]  
        [,FM=foot-margin]  
        [,HS=heading-space]  
        [,FS=footing-space]  
        [,PTSIZ=pointsize]  
        [,BLI=baseline-increment]  
        [,CP=codepage]  
        [,PGSIZ=paper size name]  
        [,OUTFILE=listfile name]
```

logical-device name

Name of the logical device as it will be given with the DEVICE option of the SCRIPT command.

physical-device name

Physical device name. For each physical device, there must be a physical device table associated with it. The physical device table describes the physical characteristics of the device.

font-name Name of the default font.

For line devices, this name must be defined with a DSMFIB macro in source file DSMTFFIB.

For page printers, this name must be a member of the device font library. Specify the coded font identifier. This value can be changed with the .BF [Begin Font] control word or with the CHARS option on the SCRIPT command.

For PostScript devices, this name must be a member of the device font library. Specify the AFM filename. This value can be changed with the CHARS option on the SCRIPT command.

The default value is "MONO."

See "Required Fonts for Page Printers" on page 151 for more information about the default font.

lpi	<p>Fixed vertical line spacing value. The value must be one of the following:</p> <table> <tr><td>6</td><td>6 lines per inch</td></tr> <tr><td>8</td><td>8 lines per inch</td></tr> <tr><td>10</td><td>10 lines per inch</td></tr> <tr><td>12</td><td>12 lines per inch</td></tr> </table> <p>This parameter is ignored for page printers and PostScript devices.</p> <p>The default value is 6. This value is used to compute the size of the vertical device unit.</p>	6	6 lines per inch	8	8 lines per inch	10	10 lines per inch	12	12 lines per inch
6	6 lines per inch								
8	8 lines per inch								
10	10 lines per inch								
12	12 lines per inch								
pitch	<p>Fixed horizontal character spacing value, given as the number of characters per inch.</p> <p>This parameter is ignored for page printers and PostScript devices.</p> <p>The default value is 10. This value is used to compute the size of the horizontal device unit.</p>								
rotation	<p>The angle of rotation of the page, measured in degrees. It applies to page printers (except the 4250) and PostScript devices. The value must be one of the following multiples of 90 degrees:</p> <table> <tr><td>0</td><td></td></tr> <tr><td>90</td><td>or -270</td></tr> <tr><td>180</td><td>or -180</td></tr> <tr><td>270</td><td>or -90</td></tr> </table> <p>The default value is 0.</p>	0		90	or -270	180	or -180	270	or -90
0									
90	or -270								
180	or -180								
270	or -90								
page-len	<p>Default page length, measured in vertical device units. This value can be changed with the .PL [Page Length] control word.</p> <p>The default value is 66.</p>								
page-wid	<p>Default page width, measured in horizontal device units. This value can be changed with the .PW [Page Width] control word.</p> <p>The default value is 132.</p>								
line-len	<p>Default line length, measured in horizontal device units. This value can be changed with the .LL [Line Length] control word.</p> <p>The default value is 60.</p>								
page-marg	<p>Default page margin, measured in horizontal device units. Either:</p> <p>PM=binding</p> <p>or:</p> <p>PM=(odd-binding,even-binding)</p> <p>can be given. If only one value is given, it applies to all pages. If two values are given, they must be enclosed in parentheses; the first applies to odd pages, and the second applies to even pages.</p> <p>This value can be changed with the BIND option of the SCRIPT command and with the .PM [Page Margins] control word.</p> <p>The default value is 0.</p>								

top-margin

Default top margin, measured in vertical device units. This value can be changed with the .TM [Top Margin] control word.

The default value is 3.

bot-margin

Default bottom margin, measured in vertical device units. This value can be changed with the .BM [Bottom Margin] control word.

The default value is 3.

heading-margin

Default heading margin, measured in vertical device units. This value can be changed with the .RH [Running Heading] control word.

The default value is 0.

footing-margin

Default footing margin, measured in vertical device units. This value can be changed with the .RF [Running Footing] control word.

The default value is 0.

heading-space

Default heading space, measured in lines. This value can be changed with the .RH [Running Heading] control word, but the heading margin plus the heading space must fit within the top margin.

The default value is 0.

footing-space

Default footing space, measured in lines. This value can be changed with the .RF [Running Footing] control word, but the footing margin plus the footing space must fit within the bottom margin.

The default value is 0.

pointsize

Default point size for PostScript devices. This value is valid only for PostScript devices.

Default value is 10.

baseline-increment

Default percentage of the point size used to adjust the baseline between output lines. This value is valid only for PostScript devices.

The default value is 110. This sets the default linespace to 110% of the point size.

codepage

Default code page to be used for PostScript devices. This value is valid only for PostScript devices.

The default is PSCP0395.

paper size name

Default paper size name for PostScript devices. This value is valid only for PostScript devices.

The default name is "letter."

listfile name

Default filetype for CMS or dataset name qualifier for MVS for listfile when the FILE command option is not specified, or when the FILE command option is specified without a name.

Default is LISTxxxx, where xxxx is the logical device requested.

Refer to the *DCF: SCRIPT/VS Language Reference* for information about how this is used with the FILE command option.

For example, the DSMLDT specification for the 1403N6 logical device provided with SCRIPT/VS is coded as:

```
DSMLDT LD=1403N6,PD=1403,  
PW=85,PL=66,PM=10
```

The line length for this logical device is the default, 60 characters (six inches), because the LL parameter was not specified. If your installation plans to use the 1403N6 logical device only to produce masters for duplexed reproduction, you might want to change its DSMLDT specification:

```
DSMLDT LD=1403N6,PD=1403,  
PW=85,PL=66,  
LL=66,PM=(13,6)
```

In this DSMLDT specification, the line length is set to 66 characters (about 40 picas). The page margins are set to 13 spaces for odd pages and 6 spaces for even pages.

If your installation has a 3800 Model 1 application that uses 8.5-inch by 11-inch paper and requires no page margins, you can define a new logical device for this application, which uses the entire form:

```
DSMLDT LD=3800F,PD=3800,DF=GSC,  
LPI=12,FS=4,  
PW=1530,PL=120,LL=1530,  
PM=0,TM=0,BM=0
```

If you want to define a 3820 Page Printer logical device that is the same as the standard 3820A device except that the default font is 10-point Sonoran Sans Serif, you could specify:

```
DSMLDT LD=3820X,PD=3820,DF=X0A00395,  
PW=2040,PL=2640,LL=1440,  
PM=240,FSS=1024,TM=120,BM=120
```

If you want to change the default filetype to LIST90 when using the 3820A90 logical device, you can change the DSMLDT specification:

```
DSMLDT LD=3820A90,PD=3820,  
PW=2040,PL=2640,LL=1440,  
PM=240,FSS=1024,TM=120,BM=120  
OUTFILE=LIST90
```

The DSMSLDT Macro

The file DSMLPLDT contains one DSMSLDT macro following the last DSMLDT macro specification.

The DSMSLDT macro is coded as follows:

DSMSLDT

This macro creates a map of the logical device table. It must be the last macro coded in DSMLPLDT and requires no parameters.

DSMLPPDT: Physical Device Tables

When documents are formatted, SCRIPT/VS takes into account the characteristics of the physical output device. DSMLPPDT defines all the physical devices that are referred to by the logical devices defined in DSMLPLDT. There is one physical device table (PDT) for each physical device. Several logical devices can refer to one physical device.

For information about the logical devices, see "DSMLPLDT: Logical Device Definitions" on page 131

Note: When adding new physical devices, you must update the GML profile (for example, DSMPROF4) to include processing for these added physical devices.

DSMPDT Macro

This macro is used by the DSMLPPDT module to create the entries to the physical device table. A physical device table describes the physical characteristics of a device.

The DSMPDT macro is coded as follows:

[label] PDT	PD = physical-device name [,OD = output-type] [,FL = fontlib] [,PSL = pseglib] [,BSS = baseline-shift] [,ROT = rotation] [,MF = maxfont] [,ICS = intercharacter-spacing] [,HR = horizontal-resolution] [,VR = vertical-resolution] [,FSS = font-size] [,PSEG = map-pseg] [,COLOR = yes no]
-------------	--

physical-device name

The physical device name as it will be used when defining a logical device.

output-type The type of output data stream that this physical device will accept. The valid output data streams that are recognized by this macro are:

Output Type	Physical Device Type
1403	1403 Line Printer (default)
2741	2741 Typewriter Terminal
3270	3270 Display Terminal
3800	3800 Printer (Model 1)
38PP	3800 Printer (Models 3 and 6)
4250	4250 Electroerosion Printer
3820	AFP Printers ⁶
PS	PostScript Device
STAIRS	STAIRS

Note: The first four physical devices listed are known as line devices; the 4250, 38PP, and 3820 are page devices; and PS refers to PostScript devices configured to accept 8-bit ASCII.

fontlib The library where fonts are to be found. This parameter only applies to page devices and PostScript devices. If a font library is specified with a line device, an error message is issued. The valid font library names recognized by DSMPDT are:

FONT3820	For the 3820 and 4224 devices
FONT38PP	For the 3800 device (Models 3 and 6)
FONT4250	For the 4250 device
FONT300	For the 4028 device
FONTPS	For the PostScript devices

If this parameter is not specified, the defaults will depend on the output data stream specified:

Physical Device Type	Default Font Library
3820	FONT3820
38PP	FONT38PP
4028	FONT300
4250	FONT4250
PS	FONTPS

⁶ This physical device type refers to the AFP printers supported by SCRIPT/VS. See "Printer References" on page 4 for a list of these printers.

The following combinations are invalid:

**Physical
Device**

Type Default Font Library

3820	Any name other than FONT3820 or FONT38PP
38PP	Any name other than FONT38PP or FONT3820
4028	Any name other than FONT300 or FONT3820
4224	Any name other than FONT3820 or FONT38PP
4250	Any name other than FONT4250
PS	Any name other than FONTPS

pseglib

The library where page segments are to be found. This parameter applies only to page devices. If a page segment library is specified with a line device or PostScript device, an error message is issued. The valid page segment library names recognized by DSMPDT are:

PSEG3820	For the 3820 and 4028 device
PSEG38PP	For the 3800 device (Models 3 and 6)
PSEG4250	For the 4250 device

If this parameter is not specified, the defaults will depend on the output data stream specified:

Output

Type Default PSEG Library

3820	PSEG3820
38PP	PSEG38PP
4250	PSEG4250

The following combinations are invalid:

Output

Type Default PSEG Library

3820	PSEG4250
38PP	PSEG4250
4250	Any name other than PSEG4250

baseline-shift

The baseline shift direction allowed by the physical device. This parameter applies only to page devices and PostScript devices, and it issues an error message if specified for line devices. Valid values for this parameter are:

none	No baseline shifts allowed.
all	Both positive and negative baseline shifts allowed.

rotation Rotations that are allowed by this physical device. These values are expressed in degrees. The valid rotations recognized by DSMPDT are: 0, 90, 180, and 270. If any other rotation is specified, an error message is issued. This parameter is specified as follows:

ROT=(0,90,180,270)

The above definition states that the physical device can rotate 0, 90, 180, and 270 degrees. If the physical device cannot rotate text, this parameter should not be specified.

Note: The rotation parameter applies only to area, page, table, and cell orientation. This parameter does not affect the character rotation you specify with the .DF [Define Font] control word.

There is a difference between not coding the 'ROT' parameter and coding 'ROT=(0)'. In areas, for example, if the 'ROT' parameter is not coded, then any rotation specified on the .DA [Define Area] control word will be ignored. This is recommended when formatting for line devices. If 'ROT=(0)' is coded, then only a composite rotation of 0 will be allowed. Any other composite rotation specified will be ignored and will result in an error.

maxfont The maximum number of fonts allowed per page. This parameter applies only to the 38PP, 3820, and 4224 physical output types. If specified, this number must be greater than zero. If not specified, the defaults are:

**Physical
Device**

Type	Maxfont
38PP	64
3820	32
4028	No maximum
4224	32

intercharacter-spacing

The direction of intercharacter spacing allowed by this physical device. This parameter applies only to page devices and PostScript devices, and issues an error message if specified for line devices. The directions that can be specified are:

pos Positive intercharacter spacing allowed.

none No intercharacter spacing allowed.

all Positive and negative intercharacter spacing allowed.

horizontal-resolution

The horizontal resolution of the device, measured in device units per inch. This value is used to compute the actual physical size of a horizontal device unit, which is used in converting horizontal space units (for example, inches) to device units. This parameter applies only to page devices and PostScript devices, and issues an error message if specified for line devices.

vertical-resolution

The vertical resolution of this device, measured in device units per inch. This value is used to compute the actual physical size of a vertical device unit, which is used in converting vertical space units

(for example, inches) to device units. This parameter applies only to page devices and PostScript devices, and issues an error message if specified for line devices.

font-size

The font raster pattern storage size (capacity) of the physical device, measured in thousands of bytes.

For line devices, this value is compared to the font size parameter values given with the DSMFIB macro in the DSMTFFIB module for all fonts specified with the CHARS option of the SCRIPT command.

For page devices, this value is compared to the sum of all the storage required for each font object used in one page of output. If not specified, the default is "1" (1K bytes), which will not be enough storage for typographic fonts.

For the LaserPrinter 4028 and PostScript devices, this value is not used.

map-pseg

Specifies if all page segments are to be mapped or not mapped by default, on a page-by-page basis. This parameter is ignored for all devices except the 3820 Page Printer and the 3800 Model 3 or 6. Valid values are either MAP or NOMAP. NOMAP is the default. MAP means that all the page segments used on a particular page are included in a Map Page Segment (MPS) structured field in the active environment group. The page segment in the MPS structured field is downloaded to the page device prior to the printing of each page. A performance savings is gained if the page segment is used more than once (for example, in the heading of each page), because it is downloaded only once.

This setting can be overridden for a particular page segment by specifying the MAP or NOMAP parameter on the .SI [Segment Include] control word.

color

Specifies whether the physical device supports color. The default is "NO."

DSMSPDT Macro

The file DSMLPPDT contains one DSMSPDT macro following the last DSMPDT macro specification.

This macro is coded as follows:

DSMSPDT

This macro creates a map of the physical device table. It must be the last macro called by the DSMLPPDT module and requires no parameters.

DSMTFFIB: Line Printer Font Information

When formatting for page printers or PostScript devices, SCRIPT/VS obtains font information, such as the widths of characters in a particular font, from a host system library. When formatting for line devices, SCRIPT/VS obtains this information from internal tables.

DSMFIB Macro

The file DSMTFFIB contains one font information table for each line device font specified with a DSMFIB macro. The 1403, 2741, and 3270 devices have no font capability; the default font, MONO, is the only font applicable to these devices. The 3800 Printing Subsystem Model 1 has a limited font capability; most of the fonts described in the file DSMTFFIB are for this device. Table 16 on page 252 lists the 3800 Printing Subsystem fonts provided with DCF; DSMTFFIB contains font information tables for each of these fonts, as well as for all of the fonts provided with the IBM 3800 Printing Subsystem.

If you change any of the fonts provided with the IBM 3800 Printing Subsystem or with DCF, you should make corresponding changes to the DSMFIB macros that describe those fonts. If you add a new font to your installation's 3800 font library, you must create a new DSMFIB macro describing the new font before you can use it with SCRIPT/VS.

The DSMFIB macro contains the following parameters:

```
DSMFIB  NAME = font-name,  
        WTABLE = wtable  
        [,BCS = bcs-name]  
        [,PITCH = pitch]  
        [,CODE = font-code]  
        [,FS = font-size]
```

font-name Name of the font. This name can be used with:

- The DSMLDT macro in the file DSMLPLDT.
- The CHARS option of the SCRIPT command.
- The FONT parameter of the .DF [Define Font] control word.

wtable Name of a width table defined in the file DSMTFFIB. The value can be one of the names listed in "Line Device Width Tables" on page 143.

bcs-name Name of a box character set to be used when drawing boxes and rules in this font. **bcs-name** must be defined with a DSMBCS macro in the file DSMLPBCS.

pitch Number of characters printed per inch, measured horizontally. The pitch of the MONO font for 1403, 2741, and 3270 devices is 10. The pitch of 3800 fonts can be 10, 12, or 15.

The default value is 10.

font-code Indicates whether the font is monospace or proportional, and whether the font has proportional blanks. (Fonts are monospace if all of the nonblank characters have the same widths.) The valid values of **font-code** are:

M Indicates a monospace font without proportional blanks. This is the default, and it is applicable to all devices.

MB Indicates a monospace font with proportional blanks. This is applicable for fonts designed for use on the IBM 3800 Printing Subsystem Model 1.

PB Indicates a proportional font with proportional blanks. Some fonts designed for use with the IBM 3800 Printing Subsystem contain both proportionally spaced characters and proportional blanks (GP12, for example). The only 3800 Printing Subsystem font distributed with DCF that is proportional is GP12.

The default value is M.

font-size Relative size of the font, compared to the font storage size of the logical device. This parameter is used in conjunction with the FSS parameter of the DSMLDT macro specified in the file DSMLPLDT to ensure that all of the fonts specified with the CHARS option of the SCRIPT command will actually fit in the device.

For 3800 Printing Subsystem fonts, this parameter gives the number of writable character generation modules (WCGMs) required.

The default value is 1.

For example, if your installation's 3800 has an APL character set, you must add a DSMFIB macro describing it to DSMTFFIB before that font can be used with SCRIPT/VS:

```
DSMFIB NAME=APL,  
        WTABLE=MONO10,  
        FS=4
```

The font APL is monospace and uses the MONO10 width table. The character set fills four WCGM buffers in the 3800 line printer.

Note that all 3800 Printing Subsystem fonts distributed with DCF are monospace with proportional blanks, except GP12. GP12 has proportionally spaced characters and proportional blanks.

```

GP12WT  DS  0D

*          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
* 00-3F:
DC H'15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15' -----
DC H'15,18,15,12,15,15,15,15,15,15,15,15,15,15,15,15' -----
DC H'15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15' -----
DC H'15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15' -----
* 40-7F:
DC H'15,15,15,15,15,15,15,15,15,15,12,15,12,15,12,15' -----¢.<(+|
DC H'18,15,15,15,15,15,15,15,15,12,15,15,12,12,15,15' &-----!$*);-
DC H'15,18,15,15,15,15,15,15,15,15,15,12,18,18,15,15' -/-----,%_>?
DC H'15,15,15,15,15,15,15,15,15,15,12,18,18,12,15,15' -----:#@'="
* 80-BF:
DC H'15,15,15,15,15,15,12,15,15,12,15,12,15,12,12,15' -abcdefghi-----
DC H'15,12,15,12,18,15,15,15,15,15,15,12,15,12,15,12' -jklmnopqr-----
DC H'12,15,15,12,15,15,18,15,15,15,15,15,15,15,12,15' --stuvwxyz-----
DC H'12,12,12,12,12,12,12,12,12,12,15,15,15,12,15,15' -----
* C0-FF:
DC H'15,15,15,15,15,15,15,15,15,12,15,15,15,15,15,15' -ABCDEFGHI-----
DC H'15,12,15,12,18,15,15,15,15,15,15,15,15,18,15,15' -JKLMNOPQR-----
DC H'18,15,15,15,15,15,18,15,15,15,15,15,15,15,15,15' --STUVWXYZ-----
DC H'15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15' 0123456789-----
*
*          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF

```

Figure 2. Example of a Font Width Table. GP12 is a 12-pitch proportionally spaced font with proportional blanks.

Line Device Width Tables

The file DSMTFFIB also contains character width tables for all line device fonts. These tables give the width of each character code point, measured in horizontal device units, and are coded as 256 consecutive halfword values.

The following tables are provided in the IBM-supplied file:

- | | |
|---------------|---|
| GP12WT | This width table is intended for use only with the 3800 line printer font GP12. |
| MONOWT | This table gives the widths of all characters as one horizontal device unit. It is intended for use by fonts for the 1403, 2741, and 3270 devices. |
| MONO10 | This table gives the widths of all characters as 18 horizontal device units. It is intended for use by 10-pitch monospace 3800 line printer fonts, such as GS10 and TN. |
| MONO12 | This table gives the widths of all characters as 15 horizontal device units. It is intended for use by 12-pitch monospace 3800 line printer fonts, such as GS12 and GF12. |
| MONO15 | This table gives the widths of all characters as 12 horizontal device units. It is intended for use by 15-pitch monospace 3800 line printer fonts, such as GS15 and GSC. |

- PROP10** This table gives the widths of all characters as 18 horizontal device units, except for the SCRIPT/VS proportional blank characters. It is intended for use by the 10-pitch 3800 fonts provided with SCRIPT/VS, such as GT10 and GB10.
- PROP12** This table gives the widths of all characters as 15 horizontal device units, except for the SCRIPT/VS proportional blank characters. It is intended for use by the 12-pitch 3800 fonts provided with SCRIPT/VS, such as GT12 and SI12.
- PROP15** This table gives the widths of all characters as 12 horizontal device units, except for the SCRIPT/VS proportional blank characters. It is intended for use by the 15-pitch 3800 fonts provided with SCRIPT/VS, such as GT15 and ST15.

When you add a new DSMFIB macro specification to DSMTFFIB to describe a new 3800 font, determine which of these tables describes the widths of the characters and identify it with the WTABLE parameter. If none of these tables matches the widths of the characters in your new font, code a new width table.

For logical devices using the 3800 physical device and monospace fonts with proportional blanks, the widths of X'11', X'12', and X'13' must be 18, 15, and 12 in the width table, respectively. The width of X'40' in the width table must be in the range of 12 to 18.

DSMTRNLT: National Language Character Definitions

The non-English spelling verification and hyphenation dictionaries provided with DCF contain a variety of accented characters. Because the code point assignments used for these characters at your installation might differ from those used in the IBM-supplied dictionaries, SCRIPT/VS provides code point translation tables.

Punctuation and word delimiter characters are removed from words before they are verified or hyphenated. Because the code point assignments used for these characters might differ at your installation, SCRIPT/VS provides a table of punctuation and word delimiter characters.

SCRIPT/VS also provides tables for sorting index entries. Because a hexadecimal sort is not adequate, these tables are used to define the correct order in which to sort characters in index entries. SCRIPT/VS provides separate tables for each IBM-supplied language, because each language has slightly different sort-sequence requirements.

DSMACTT Macro

The source file DSMTRNLT contains one DSMACTT macro specification for the accented-character translation tables.

Words in the non-English IBM-supplied spelling verification and hyphenation dictionaries use the accented-character code point assignments shown in Table 4 on page 90. If the code points used for these characters at your installation differ from these, you should modify the DSMACTT macro specification to define the correspondence between the external and internal representations of each character that is different.

The DSMACTT macro is specified as:

DSMACTT [(char-pair)]
[,...]

char-pair Character-pair specification, giving the external and internal code point assignments for an accented character. The **char-pair** is specified as:

(external,internal)

The DSMACTT macro can have as many character-pair specifications as necessary: one for each accented character whose internal and external representations differ.

external External representation of an accented character, specified either as a quoted character, such as '?', or as two hexadecimal digits, such as 6F.

internal Internal representation of an accented character, specified either as a quoted character, such as '?', or as two hexadecimal digits, such as 6F.

For example, suppose your installation uses the following accented characters with the external code point assignments listed:

Code point	Accented Character
C0	A UMLAUT (LOWERCASE)
4A	A UMLAUT (UPPERCASE)
6A	O UMLAUT (LOWERCASE)
E0	O UMLAUT (UPPERCASE)
D0	U UMLAUT (LOWERCASE)
5A	U UMLAUT (UPPERCASE)
A1	ESS ZET

Using internal code point assignments, you should code the DSMACTT macro as follows:

	Column 16	Column 72
DSMACTT	(C0,43),(4A,63),	*
	(6A,CC),(E0,EC),	*
	(D0,DC),(5A,FC),	*
	(A1,59)	

Note: When coding the DSMACTT macro, if multiple lines are required, you must put an asterisk in column 72 of all but the last line, and start all but the first line in column 16.

Refer to "Communicating with SCRIPT/VS" in the *DCF: SCRIPT/VS User's Guide* for additional information about accented-character code points.

DSMWDPT Macro

The source file DSMTRNLT contains DSMWDPT macro specifications for word-delimiter and punctuation tables for each language.

For spelling verification and hyphenation, the hyphen (-) and single quotation mark (') are considered punctuation. Characters that are considered word delimiters are listed in Chapter 8, "Verifying Spelling" on page 85.

The DSMWDPT macro is specified as:

```
DSMWDPT (char-list) [...],  
        LANG=lang-code
```

char-list List of characters that are to be considered word delimiters or punctuation characters. The **char-list** is coded as:

(key,char<,...>)

As many character list specifications as necessary can be coded.

key WD for word delimiters or PUNC for punctuation characters.

char Character, specified either as a quoted character, such as "?", or as two hexadecimal digits, such as 6F. As many character specifications as necessary can be coded.

lang-code Indicates which language this table applies to. The valid codes are:

DAN	Danish
DUTH	Dutch
EAM	American English
ECAN	Canadian English
EUK	United Kingdom English
FCAN	Canadian French
FIN	Finnish
FNAT	French National
GERM	German
ICE	Icelandic
ITAL	Italian
NOR	Norwegian
POR	Portuguese
SPAN	Spanish
SWE	Swedish

If the table applies to more than one language, the list of codes must be enclosed in parentheses.

For example, in the example of the DSMACTT macro shown above, the hexadecimal code points 4A and 5A are assigned to accented characters. Because these code points are normally the cent-sign (¢) and exclamation point (!), respectively, they are coded as word delimiters in the IBM-supplied DSMTRNLT file. To remove them from the word delimiter and punctuation table for German, remove the language code GERM from the DSMWDPT macro specification and code a new DSMWDPT macro:

	Column 16	Column 72
DSMWDPT	(WD,05,16),	
	(WD,11,12,13),	*
	(WD,40,41),	*
	(WD,'.',',',''),	*
	(WD,'(','(',')'),	*
	(WD,AD,BD),	*
	(WD,8B,9B),	*
	(WD,AF,BF),	*
	(WD,'_','?',',',''),	*
	(WD,':',';','','\$'),	*
	(WD,'/','+','*'),	*
	(WD,'=',' ','~'),	*
	(WD,'<','>','%'),	*
	(PUNC,'-',',7D),	*
	LANG=GERM	

Note: When coding the DSMWDPT macro, if multiple lines are required, you must put an asterisk in column 72 of all but the last line, and start all but the first line in column 16.

DSMHHCC Macro

The DSMACTT and DSMWDPT macros use the DSMHHCC macro. DSMHHCC is not coded in DSMTRNLT.

Index Sort Sequence Macros

The DSMTRNLT module calls the DSMIXTT1 and DSMIXTT2 macros to build translation tables. These tables are used to translate a given string to its equivalent sort sequence value. DSMIXTT2 builds a 1-to-2 table. DSMIXTT1 builds a 1-to-1 table.

Using the tables is a two-step process. The 1-to-2 table translates single characters to the 1- or 2-character string that will be used when sorting that character.

The 1-to-1 table is used to translate each character to the 1-byte sort-sequence value for that character.

DSMIXTT1 and DSMIXTT2 generate modules that are named in the following format:

- DSMxxXT1
- DSMxxXT2.

where xx is the first 2 characters of the language code. For example, DSMDUXT1 represents Dutch. See Chapter 13, "Index Sort Sequence" on page 177 for more information on modifying sort-sequence tables.

DSMIXTT2 Macro

The DSMTRNLT module calls the DSMIXTT2 macro to create the 1-to-2 index sort-sequence tables. The 1-to-2 tables created by this macro are used to define the 2-character expansion for the special single characters that should be sorted as a 2-character string. For example, the character "ß" is translated to the character string "ss" and will be sorted as "ss."

The 1-to-2 tables are also used to define the header character for any character that sorts under a header that is not that character. The header character is

what appears in the index to indicate a change to another index group. For example, if “Á” sorts under “A,” then “Á” is in the 1-to-2 table as “Á” translated to “A.” Lowercase letters are translated to the equivalent uppercase letters using this table.

The DSMIXTT2 macro is specified as:

```
DSMIXTT2 (char-pair)
    [...]
```

```
LANG=lang-code
```

char-pair Specifies a character and the 1- or 2-character string it should be converted to. As many **char-pair** specifications as necessary can be entered. The format of a **char-pair** specification is:

(char,charstr)

char The external representation of the character to be converted. It can be specified as either the actual character value or the hexadecimal value. If the actual character value is used, it must be enclosed in single quotation marks. The following example shows the two ways to enter the **char** value:

('A',charstr)
(C1,charstr)

charstr The 1- or 2-character string that the first character should be converted to. It can be specified as either the actual character value or with two or four hex digits. If the actual character value is used, it must be enclosed in single quotation marks. The following example shows the two ways to enter a 1-character **charstr** value:

(char, 'A')
(char,C1)

The following example shows the two ways to enter a 2-character **charstr** value:

(char, 'AE')
(char,C1C5)

lang-code Indicates which language or language this table applies to. The valid codes for these languages are:

DAN	Danish
DUTH	Dutch
EAM	American English
ECAN	Canadian English
EUK	United Kingdom English
FCAN	Canadian French
FIN	Finnish
FNAT	French National
GERM	German
ICE	Icelandic
ITAL	Italian
NOR	Norwegian

POR	Portuguese
SPAN	Spanish
SWE	Swedish

If more than one language is specified, the language names must be separated by commas, and the entire list of names must be enclosed in parentheses.

Note: When coding the DSMIXTT2 macro, if multiple lines are required, you must put an asterisk in column 72 of all but the last line, and start all but the first line in column 16.

DSMIXTT1 Macro

The DSMTRNLT module calls the DSMIXTT1 macro to create the 1-to-1 index sort-sequence tables. The 1-to-1 tables created by this macro are used to define the correct sort sequence value for a given character. The DSMIXTT1 macro is specified as:

```
DSMIXTT1 (char-pair)
        [... ]
        LANG=lang-code
```

char-pair	Specifies a character and the index sort value assigned to that character, which will be used when sorting the character. The format of char-pair is: (char,sortval)												
char	The external representation of the character to be converted. It can be specified either as the actual character value or as two hexadecimal digits. If the actual character value is used, it must be enclosed in single quotation marks. The following example shows the two ways to enter the char value: ('A',sortval) (C1,sortval)												
sortval	The value assigned to the character for sorting. It can be specified either as the actual character value or with two hexadecimal digits. If the actual character value is used, it must be enclosed in single quotation marks. The following example shows the two ways to enter a 1-character sortval value: (char, 'A') (char, C1)												
lang-code	Indicates which language or languages this table applies to. The valid codes for these languages are: <table> <tr> <td>DAN</td><td>Danish</td></tr> <tr> <td>DUTH</td><td>Dutch</td></tr> <tr> <td>EAM</td><td>American English</td></tr> <tr> <td>ECAN</td><td>Canadian English</td></tr> <tr> <td>EUK</td><td>United Kingdom English</td></tr> <tr> <td>FCAN</td><td>Canadian French</td></tr> </table>	DAN	Danish	DUTH	Dutch	EAM	American English	ECAN	Canadian English	EUK	United Kingdom English	FCAN	Canadian French
DAN	Danish												
DUTH	Dutch												
EAM	American English												
ECAN	Canadian English												
EUK	United Kingdom English												
FCAN	Canadian French												

FIN	Finnish
FNAT	French National
GERM	German
ICE	Icelandic
ITAL	Italian
NOR	Norwegian
POR	Portuguese
SPAN	Spanish
SWE	Swedish

If more than one language is specified, the language names must be separated by commas, and the entire list of language names specified must be enclosed in parentheses.

Note: When coding the DSMIXTT1 macro, if multiple lines are required, you must put an asterisk in column 72 of all but the last line, and start all but the first line in column 16.

Replacing the IBM-Supplied Dictionaries

You can have SCRIPT/VS use a dictionary you create instead of the dictionaries supplied by IBM. To create your own dictionary, you need to know the two-letter language code for your language. The following list shows those codes:

DA	Danish
DU	Dutch
EA	English American
EC	English Canadian
EU	English U.K.
FC	Canadian French
FN	French National
GE	German
IC	Icelandic
IT	Italian
NO	Norwegian
PO	Portuguese
SP	Spanish
SW	Swedish

Use the following steps to replace the IBM-supplied dictionary for your language with your own dictionary:

1. Create your own dictionary using the steps outlined in Chapter 11, "Creating and Maintaining User Dictionaries" on page 157.
2. Two of the files created will be named "DSMD****" and "DSMH****," where "****" is the name of your DMPIN files. You need to make the following modifications to change the name of those files to DSM##DRM and DSM##HRM, where "##" is the two-letter language code listed above. The first line in your "DSMD****" file looks something like:

```
DSMD*** CSECT
```

Change this to:

```
DSM##DRM CSECT
```

3. The last line in your "DSMD****" file looks something like:

END DSMD***

Change this to:

END DSM##DRM

4. The first line in your "DSMH***" file looks something like:

DSMH*** CSECT

Change this to:

DSM##HRM CSECT

5. The last line in your "DSMH***" file looks something like:

END DSMH***

Change this to:

END DSM##HRM(1)

6. Rename the "DSMD*** ASSEMBLE" file to "DSM##DRM ASSEMBLE"

7. Rename the "DSMH*** ASSEMBLE" file to "DSM##HRM ASSEMBLE"

8. Compile both modules.

9. If your dictionaries are link edited into your load module, link edit the text decks created into your load module again.

10. Put these two text decks into your text library (DSMHYLIB TXTLIB).

Required Fonts for Page Printers

SCRIPT/VS requires special fonts to produce output for page printers. The defaults chosen by SCRIPT/VS and the GML starter set require the following font licensed programs.

For the IBM LaserPrinter 4028:

- Courier font metrics
- Times Roman font metrics

Note: These fonts are available in PSF Version 2 (MVS and VM).

For the IBM 4224 Printer:

- Courier
- Essay

Note: These fonts are included with PSF.

For the 4250 Printer:

- 5771-AAR: Monotype Times New Roman
- 5771-AAW: Typewriter and Pi.

For other IBM AFP page printers (such as the 3820):

- 5771-ABA Sonoran Serif
- 5771-ABC Pi and Specials
- Prestige fonts available with the PSF and PMF licensed programs.

You can modify DCF at installation time or use the CHARS option of the SCRIPT command to point to typeface families other than the required ones listed here.

The GML starter set uses typewriter fonts, Pi and Specials, and various point sizes and styles of the initial font.

The defaults are chosen by DSMLPLDT and the GML starter set (refer to the *DCF: GML Starter Set Implementation Guide* for more details on how the defaults are chosen in GML).

If you choose to use a typewriter-like, monospaced font (such as one of the PSF Prestige fonts for the 3820 or 3800 Model 3) as the default, you will need to modify the GML profile, DSMPROF4. Refer to the *DCF: GML Starter Set Implementation Guide* for an example of modifying DSMPROF4 to support a monospaced, default font.

In DSMLPLDT, the DF option of the DSMLDT macro defines a default coded font that points to a font (also called a character set) and a code page. For example, for the 3800 Model 3 logical devices, DF=X1T00395 points to:

- C1T05500: 0 degree 10 point Sonoran Serif
- T1GI0395: US, Canadian (English) Code Page.

for the IBM LaserPrinter 4028 logical device, DF=X0E21T0C points to:

- C0E20T00: 0 degree 10 point Times Roman medium
- T1VI0500: International #5

and for the 4250 logical devices, DF=AFTTR395 points to:

- AFT06008: 10 point Monotype Times New Roman
- AFTC0395: US, Canadian English Code Page

If you want the default to point to some other font or code page, you must do the following:

1. Install the appropriate font licensed program.
2. Select the coded font that you want to use (see "DSMLPLDT: Logical Device Definitions" on page 131 for more details on how to modify the DSMLPLDT file).
3. Run the Font Library Index Program.

For page printers, the correct font licensed programs must be installed and the Font Library Index Program run to create a required index. See Appendix D, "The DCF Font Library Index Program" on page 261 for more details on how to use the Font Library Index Program.

For the 3800 Printing Subsystem Model 3, the font objects used by SCRIPT/VS (and any others your installation will be using) must be created according to the procedures described in the following publications:

- *ABOUT TYPE: IBM's Guide for Using Print Management Facility with Type*
- *Print Management Facility User's Guide and Reference*
- *Font Library Service Facility Installation and Operation.*

For the 3820 Page Printer, the font objects used by SCRIPT/VS (and any others your installation will be using) must be created according to the procedures described in the following publications:

- *ABOUT TYPE: IBM's Guide for Using Print Management Facility with Type*
- *Print Management Facility User's Guide and Reference*
- *Font Library Service Facility Installation and Operation.*

Required Fonts for PostScript Devices

To use the unmodified starter set with PostScript devices, you must make Times Roman and Courier typeface families available to your PostScript device.

See "PostScript Font Support" on page 277 for a description of how DCF handles PostScript fonts.

Chapter 11. Creating and Maintaining User Dictionaries

Chapter 11. Creating and Maintaining User Dictionaries	157
Dictionary Maintenance	157
Sample Procedure	160
MVS PROC Defaults	161
Table of STEPS, DDNAMES, and DSNs for the DM PROC	162

Chapter 11. Creating and Maintaining User Dictionaries

You can create dictionaries to provide spelling verification and hyphenation support for words that are frequently used in your documents but are not included in the SCRIPT/VS main dictionary. These words may reflect the nature of your business and may include such things as technical terms and company abbreviations and acronyms.

The IBM-supplied dictionaries for the following languages are empty: Danish, Finnish, Icelandic, Norwegian, Portuguese, and Swedish. However, user dictionaries can still be created for those languages.

Dictionary Maintenance

User dictionaries are created during dictionary maintenance. At that time, words that are being placed in the user dictionary are reduced to a numerical representation, which is expressed in Assembler language source statements. These source statements are assembled and linked to produce a module that is loaded by SCRIPT/VS when a .DL [Dictionary List] control word that refers to the user dictionary is encountered.

Three types of reports can be generated during the dictionary-maintenance process:

- The alpha dictionary report, which is an alphabetical list of all words in the Alpha Dictionary file
- The hyphenation report, which is an alphabetic list of all words in the Alpha Dictionary file with their hyphenation points shown
- The oldest-usage report, which lists words that have not been frequently updated during dictionary maintenance.

Dictionary maintenance is run as a batch job with the appropriate JCL. A sample of the JCL/JCS needed to create a user dictionary in an MVS or VSE environment is provided with SCRIPT/VS on the base materials installation tapes. An EXEC is provided for dictionary maintenance in the CMS environment.

The dictionary-maintenance process requires an input file that consists of a control card and a record of dictionary-update transactions:

- The control card indicates which of the three reports, if any, are to be produced and the type of updates that are to be made to the Alpha Dictionary file. Table 5 on page 159 illustrates the format of this control card.
- The update transaction records contain the words that are to be added to or deleted from the user dictionary. Table 6 on page 160 illustrates the format of these records.

The control card and the update transaction records are all 80 characters long. The information in the control card and the first two bytes of the update transactions must be entered in all uppercase letters.

During spelling verification, words are checked against the SCRIPT/VS main dictionary and against the words in any dictionaries that you have created and indicated using the .DL [Dictionary List] control word.

When user dictionaries are used to verify words in a document, words are recognized as valid depending on how they are entered in the update transaction records. Therefore, when placing words in the update transaction records, remember the following:

- Words entered with all uppercase letters are considered valid only when they appear in all uppercase.
- Words that start with a capital letter are considered valid when they appear either in all uppercase or start with a capital letter.
- Words entered with all lowercase letters are considered valid when they appear in all uppercase letters, all lowercase letters, or start with a capital letter.
- Words that contain both upper and lowercase letters, such as McDonald, are considered valid only when they appear exactly as entered.

Table 5. Dictionary-Maintenance Process Control Card Format		
Byte	Field	Description
1	BLANK	This field is reserved.
2	REPORT	Indicates which reports are to be generated: blank no reports 1 both reports 2 the alpha dictionary report 4 the hyphenation report 5 the alpha dictionary and oldest-usage reports 6 the alpha dictionary and hyphenation reports 7 the oldest-usage and hyphenation reports
3 – 5	BLANK	This field is reserved.
6 – 7	OLDUSG	Contains a 2-digit number that is decremented by the number specified in the OUDEC field each time the dictionary-maintenance process is run. When this number becomes less than or equal to the value specified in the OUDEC field, the temporary word it applies to is eligible for deletion if the Alpha Dictionary file becomes too large. This field applies only for nonpermanent words.
8 – 9	OUDEC	Contains a 2-digit number (nn) that is used to determine when a nonpermanent word is eligible for deletion from the Alpha Dictionary file.
10	BLANK	This field is reserved.
11	DELALP	Indicates that any temporary word whose oldest-usage counter (field OLDUSG) is below the number specified in the DELSET field is to be deleted. (D is specified in this case.) Leave this field blank if you do not want to delete infrequently used words.
12 – 14	DELSET	Specifies a 3-digit number that is used in conjunction with the DELALP field to determine when a nonpermanent word should be deleted.
15 – 20	MAXDICT (in bytes)	Contains a six-digit number that indicates the maximum size of the user dictionary (usually 100000 bytes). If left blank, the default is to 100000.
21 – 26	RUNDATE	Indicates (in the form mmddyy) the date that the dictionary-maintenance process is being run. This date is placed in the DATE field of the record for each word being added if the field was left blank.
27 – 70	BLANK	This field is reserved.
71 – 72	CONTROL	Contains the characters "TR," which indicate that this is a control card.

Table 6. Format of an Update Transaction Record		
Bytes	Field	Description
1	ACTION	Contains the code that indicates what action is to be performed: A indicates that the word is to be added to the dictionary. H indicates that the word includes its hyphenation points. D indicates that the word is to be deleted. The action code must be an uppercase letter.
2	PERM	Indicates whether the word is to be permanent or temporary. P is specified if the word is permanent and can be deleted only by specifying a D transaction code; it is left blank if the word is temporary and can be deleted if not used for a set period of time (it can also be deleted by specifying the D action code); and @ is specified if the status of the word is to be changed from permanent to temporary.
3 – 65	ALPHA	Contains the word that is to be placed in the alpha dictionary file. This word must not start or end with a hyphen, and it must consist of alphabetic characters or an apostrophe. (Hyphens can be included if H was specified in the ACTION field.) Imbedded blanks are not allowed.
66 – 71	DATE	Specifies the date that the word was added to the alpha dictionary file. The date must be specified in the form mmddyy. It defaults to the date on the control card.
72 – 80	BLANK	This field is reserved.

The update input file is checked for syntax errors and duplication. Any syntactically incorrect or duplicate transaction records are eliminated. The remaining records are then used to update the alpha dictionary file.

Sample Procedure

A sample procedure to create and maintain a user dictionary in each environment that DCF runs in is provided in the sample material on the distribution tape.

In CMS, the sample procedure is an EXEC named DSMDMP4. A description of the input required for this EXEC is produced by entering "DSMDMP4 ?".

For VSE users, sample JCS to create a user dictionary is provided on the distribution tape. DSMDMJCS shows how to create a user dictionary. DSMD2JCS shows how to link-edit the user dictionary. There are no symbolic parameters as in MVS, so almost every JCS card would have to be modified to suit the particular user. The MVS DD names in the following table correspond to the VSE DLBL names. The only real difference for VSE users is the use of the UPSI switch to indicate the first run when the input ADC and AD files should be dummy.

For MVS users, a sample PROC, DSMDMJCL, is provided and may be cataloged in the system PROCLIB or used as an instream procedure. TSO and ATMS users use the batch procedure supplied on the basic tape.

The user-provided control card determines what reports the user wants printed, so the PROC is designed to run all 14 steps, regardless. Steps 13 and 14 are long-running and are optional. If the user does not want the Oldest-Usage Report, then steps 13 and 14 can be skipped. If the user does not want a Hyphenation Report, then step 5 may be skipped.

Two significant data sets are created as a result of this procedure. They are the Alpha Dictionary Control (ADC) file and the Alpha Dictionary (AD) file. The ADC file is a one-record control file containing such information as dates, dictionary size, and so on. The AD file contains all the words. These data sets are created as a result of the first run and would normally be used as input to any subsequent runs. If you intend to run this often, making updates and so on, under MVS these two files should be set up as Generation Data Groups (GDGs). In this way you can specify generation level (0) for the input file and generation (+1) for the output. You would never have to change the PROC after the first run.

MVS PROC Defaults

The following variables are provided as symbolic parameters on the PROC statement of the MVS PROCS. The PROC assumes that all data sets provided with a symbolic parameter are cataloged.

<i>Parameter</i>	<i>Default</i>																												
A	T Provides the SYSOUT class for all reports and message files.																												
SYSOUT	SYSDA Provides the UNIT parameter for all temporary and sort work files.																												
UDRM	USER Specifies the name of the user dictionary to be created. This name is the same as that used with the .DL control word when this dictionary is required. This parameter is concatenated with the string 'DSMD' and is substituted in the SYSLMOD DD card in the link-edit step to form the loadmodule name of the user dictionary.																												
LANG	DM Specifies the language of the dictionary you want to create. The language used when a dictionary is created must match the language used by SCRIPT/VS when words are checked in the user dictionary (that is, if you build a dictionary with LANG=SP then the .DL control word must be .DL SPAN USER).																												
	<table> <tr> <th><i>Value</i></th><th><i>Language</i></th></tr> <tr> <td>DM</td><td>DAN - Danish</td></tr> <tr> <td>DM</td><td>All English languages (EAM, EUK, ECAN)</td></tr> <tr> <td>DU</td><td>DUTH - Dutch</td></tr> <tr> <td>FC</td><td>FCAN - Canadian French</td></tr> <tr> <td>DM</td><td>FIN - Finnish</td></tr> <tr> <td>FN</td><td>FNAT - French National</td></tr> <tr> <td>GE</td><td>GERM - German</td></tr> <tr> <td>DM</td><td>ICE - Icelandic</td></tr> <tr> <td>IT</td><td>ITAL - Italian</td></tr> <tr> <td>DM</td><td>NOR - Norwegian</td></tr> <tr> <td>DM</td><td>POR - Portuguese</td></tr> <tr> <td>SP</td><td>SPAN - Spanish</td></tr> <tr> <td>DM</td><td>SWE - Swedish</td></tr> </table>	<i>Value</i>	<i>Language</i>	DM	DAN - Danish	DM	All English languages (EAM, EUK, ECAN)	DU	DUTH - Dutch	FC	FCAN - Canadian French	DM	FIN - Finnish	FN	FNAT - French National	GE	GERM - German	DM	ICE - Icelandic	IT	ITAL - Italian	DM	NOR - Norwegian	DM	POR - Portuguese	SP	SPAN - Spanish	DM	SWE - Swedish
<i>Value</i>	<i>Language</i>																												
DM	DAN - Danish																												
DM	All English languages (EAM, EUK, ECAN)																												
DU	DUTH - Dutch																												
FC	FCAN - Canadian French																												
DM	FIN - Finnish																												
FN	FNAT - French National																												
GE	GERM - German																												
DM	ICE - Icelandic																												
IT	ITAL - Italian																												
DM	NOR - Norwegian																												
DM	POR - Portuguese																												
SP	SPAN - Spanish																												
DM	SWE - Swedish																												
SORTLIB	SYS1.SORTLIB Specifies the name of the library that contains a SORT program.																												

Parameter	Default
PROCLIB	SYS1.PROCLIB Specifies the name of a library that contains the SORT control cards provided in the sample material with the basic distribution tape. The members are: (MSORTEDT,MSORTAD,MSORTVEC,MSORTDRT).
PACK	USRPAK Specifies the name of a volume that contains the new AD and ADC files.
MAC & MAC1	SYS1.MACLIB Specifies the name of the system macro library used for assembling the DRM and HRM files.
LOAD	DCF.R40.DCFLOAD Specifies the name of the library that contains the new user dictionary loadmodule after the link-edit step. This data set must be available (via JOBLIB, STEPLIB, and so on) when SCRIPT/VS processes the .DL control word specifying this user dictionary.
INPUT	NULLFILE Specifies the name of the file that contains the Alpha Dictionary control card and input transactions.
IADC	NULLFILE Specifies the name of the Alpha Dictionary control input file. For the first run this should default to NULLFILE. If this data set is set up as a GDG, then on subsequent runs you can specify generation level (0).
IAD	NULLFILE Specifies the name of the Alpha Dictionary input file. For the first run this should default to NULLFILE. If this data set is set up as a GDG, then on subsequent runs you can specify generation level (0).
OADC	NULLFILE Specifies the name of the <i>new</i> Alpha Dictionary Control input file. This file is created each run and resides on the volume specified in the PACK parameter. If this data set is set up as a GDG, then you can specify it as generation level (+ 1).
OAD	NULLFILE Specifies the name of the <i>new</i> Alpha Dictionary input file. This file is created each run and resides on the volume specified in the PACK parameter. If this data set is set up as a GDG, then you can specify it as generation level (+ 1).

Table of STEPS, DDNAMES, and DSNs for the DM PROC

The following table describes each step in the dictionary build process and the files used in the various steps.

Table 7 (Page 1 of 3). Step/DDname Xref				
Step Name	DDname	DSN	Symbolic Parameter	Description
STEP1				Edit user transactions.
	UCFILE	&INPUT	INPUT	User transaction input file. (May be overridden as //STEP1.UCFILE DD ^)
	EDTFILE	&&SRTIN1		Temporary sort file.
	PRT		A	Edit diagnostic report. (SYSOUT).
STEP2				Sort edited user transactions
	SORTIN	&&SRTIN1		Input edited user transactions.
	SORTOUT	&&SRTOUT1		Sorted edited user transaction file.
STEP3				Merge edited user transactions.
	EDTFILE	&&SRTOUT1		Input sorted-edited user transaction file.
	ADCFILE	NULLFILE	IADC	Input Alpha Dictionary Control file. (For the first run, this file should default to NULLFILE. Subsequent runs can use Generation (0).)
	NACFILE	&&NACFILE		Temporary Updated Alpha Dictionary Control file (used in subsequent steps).
	ADTFILE	&&ADTFILE		Merged edited user transaction file.
	PRT		A	Merge diagnostic report. (SYSOUT).
STEP4				Update Alpha Dictionary file.
	ADTFILE	&&ADTFILE		Input merged-edited user transaction file.
	ADCFILE	&&NACFILE		Temporary Alpha Dictionary Control file (from STEP3).
	NACFILE	&&ADCFILE		Temporary Updated Alpha Dictionary control file (used in subsequent steps).
	VECFILE	&&SRTIN3		Output word vector file.
	ADFILE	NULLFILE	IAD	Input Alpha Dictionary file. (For the first run, this file should default to NULLFILE. Subsequent runs can use Generation (0).)
	NADFILE	NULLFILE	OAD	Output Updated Alpha Dictionary file. (This can be specified as Generation (+ 1).)
	NDTFILE	&&DRTFILE		Output DRM and HRM table file.
	PRT		A	Update diagnostic report. (SYSOUT).
	REPORT		A	Alpha dictionary report. (SYSOUT).
STEP5				Print hyphenation report.
	ADFILE	&OAD	OAD	Input Updated Alpha Dictionary file.
	REPORT		A	Hyphenation report. (SYSOUT).
STEP6				Sort word vector file.
	SORTIN	&&SRTIN3		Input word vector file.
	SORTOUT	&&SRTOUT3		Sorted word vector file.

Table 7 (Page 2 of 3). Step/DDname Xref				
Step Name	DDname	DSN	Symbolic Parameter	Description
STEP6A				Fill dummy word vectors.
	VECFIL	&&SRTOUT3		Sorted word vector file.
	VECFIL1	&&VECFIL3		Temporary updated word vector file.
	ADCFIL	&&ADCFIL		Temporary Updated Alpha Dictionary Control file.
STEP6B				Sort updated word vector file.
	SORTIN	&&VECFIL3		Input updated word vector file.
	SORTOUT	&&SRTOUT4		Sorted updated word vector file.
STEP7				Build DRM and HRM tables.
	ADCFIL	&&ADCFIL		Temporary Alpha Dictionary Control file (from STEP4).
	NACFIL	NULLFIL	OADC	Output Updated Alpha Dictionary Control file. (Can be specified as Generation (+ 1).)
	VECFIL	&&SRTOUT4		Input sorted-updated word vector file.
	DRTFIL	&&DRTFIL		Input DRM and HRM table file.
	NDTFIL	&&SRTIN4		Updated DRM and HRM table file.
	PRT		A	Table-build diagnostic report (SYSOUT)
STEP8				Sort DRM and HRM file.
	SORTIN	&&SRTIN4		Input Updated DRM and HRM table file.
	SORTOUT	&&SRTOUT5		Sorted Updated DRM and HRM table file.
STEP9				Build DRM and HRM assemble files.
	DRTFIL	&&SRTOUT5		Sorted updated DRM and HRM table file.
	DRMFIL	&&DRMFIL		Output DRM assemble file.
	HRMFIL	&&HRMFIL		Output HRM assemble file.
STEP10				Assemble the DRM file.
	SYSIN	&&DRMFIL		Input DRM assemble file.
	SYSGO	&&OBJ		Output assembled DRM.
STEP11				Assemble the HRM file.
	SYSIN	&&HRMFIL		Input HRM assemble file.
	SYSGO	&&OBJ		Output assembled HRM. (This data set is used (DISP=MOD) to place the HRM object file immediately behind the DRM object file where it must be.)
STEP12				Link-edit the assembled DRM and HRM files.
	SYSLIN	&&OBJ		Input assembled DRM and HRM files.
	SYSLOAD	&LOAD	LOAD & UDRM	Output user dictionary load module. (This module is used by SCRIPT/VS when a .DL control word is encountered.)

Table 7 (Page 3 of 3). Step/DDname Xref				
Step Name	DDname	DSN	Symbolic Parameter	Description
STEP13				Sort Alpha Dictionary file.
	SORTIN	&&OAD		Input Updated Alpha Dictionary file.
	SORTOUT	&&SRTOUT2		Sorted Alpha Dictionary file (for the oldest-usage report step).
STEP14				Print oldest-usage report.
	ADFILE	&&SRTOUT2		Input Sorted Alpha Dictionary file.
	ADCFIL	&&ADCFIL		Temporary Alpha Dictionary control File (from STEP4).
	REPORT		A	Oldest-usage report (SYSOUT).

A sample EXEC for CMS and sample JCS for MVS and VSE are available on the distribution tape.

Once a user dictionary is created, you can issue a .DL control word to indicate to SCRIPT/VS that this dictionary is to be used, along with the SCRIPT/VS main dictionary, for hyphenation and spelling-verification processing.

See "Replacing the IBM-Supplied Dictionaries" on page 150 for a method of substituting your own dictionary for the IBM-supplied dictionary.

Chapter 12. National Use Considerations

Chapter 12. National Use Considerations	169
Language Attribute on the :GDOC Tag	169
Index Sort Sequence	169
Spelling Verification and Hyphenation Dictionaries	170
Accented Characters	171
Message Translation	171
Default Language	172
Default Font for Page Mode Devices	172
Katakana	173
Uppercase String Translation	173
Other Considerations	173
PostScript Code Pages	173

Chapter 12. National Use Considerations

SCRIPT/VS supports several languages other than English with the following features:

- Language attribute on the :GDOC tag
- Index sort sequences
- Spelling verification and hyphenation dictionaries
- Accented characters
- Message translation
- Uppercase string translation.

Language Attribute on the :GDOC Tag

To simplify national language translation, the LANGUAGE attribute has been added to the :GDOC tag in the GML starter set. Any of 15 language parameters can be specified with the LANGUAGE attribute. The parameters are for the following languages:

Canadian English
Canadian French
Danish
Dutch
English (United Kingdom)
English (United States)
Finnish
French National
German
Icelandic
Italian
Norwegian
Portuguese
Spanish
Swedish

The LANGUAGE attribute causes the literals and message translations, spelling verification and hyphenation dictionaries, algorithmic hyphenators, and index sorting sequences to be used for the specified language.

Note: Using the LANGUAGE attribute on the :GDOC tag changes the language used for that file only. It does *not* change the default language or the resident dictionary. To change the default language and the resident dictionary see Chapter 10, "Tailoring SCRIPT/VS for Your Installation" on page 111.

Index Sort Sequence

SCRIPT/VS provides two index sort-sequence tables for each language. These tables are used to sort the index entry keys according to the conventions of each IBM-supplied language. These languages are:

Danish
Dutch
English (Canadian)
English (U.K.)

English (U.S.)
 Finnish
 Canadian French
 French National
 German
 Icelandic
 Italian
 Norwegian
 Portuguese
 Spanish
 Swedish

All of the special characters are sorted at the beginning of the index, followed by the numeric characters. The alphabetic characters are sorted last. There are a few characters, such as the hyphen and the apostrophe, which are "omitted" characters. They sort the same as a required-blank character.

Special requirements of each language are also supported. For example, in the German sort sequence, the characters "Ä," "Ö," and "Ü" sort as separate characters following the unaccented "A," "O," and "U" characters.

The sort-sequence tables can be modified by the system-support person at your installation. Chapter 13, "Index Sort Sequence" on page 177 describes how SCRIPT/VS sorts index entries (based on the language specified) and "DSMTRNLT: National Language Character Definitions" on page 144 describes how to change the index sort sequences.

Spelling Verification and Hyphenation Dictionaries

SCRIPT/VS provides root word dictionaries for use in spelling verification and hyphenation in the following languages:

Canadian French
 Danish⁷
 Dutch
 English (Canadian)
 English (U.K.)
 English (U.S.)
 Finnish⁷
 French National
 German
 Icelandic⁷
 Italian
 Norwegian⁷
 Portuguese⁷
 Spanish
 Swedish⁷

⁷ This IBM-supplied dictionary is empty. A user dictionary can be created for this language. See "Replacing the IBM-Supplied Dictionaries" on page 150 for a method of substituting your own dictionary for the IBM-supplied dictionary.

In addition, algorithmic hyphenators are provided for all these languages. For additional information about creating an algorithmic hyphenator, see Appendix G, "Algorithmic Hyphenation Interface in DCF" on page 313 and to Chapter 8, "Verifying Spelling" on page 85.

Accented Characters

Words in the spelling verification and hyphenation dictionaries are represented in the Multilingual Code Point Set. This is identical to the Invariant Character Set plus the International line of the proposed EBCDIC Multilingual and Multisymbol standard, except for the OE digraph, which does not appear in the new standard.

To accommodate the variety of nonstandard-keyboard character code assignments in common use for various accented characters, all words are translated with a pair of character translation tables before and after spelling verification or hyphenation is performed. These tables are contained within a module; DSMTRNLT, which is distributed in source form. See Chapter 10, "Tailoring SCRIPT/VS for Your Installation" on page 111 for a description of the DSMTRNLT module.

The lowercase and uppercase code points of characters can be paired with the .TU [Translate Uppercase] control word to ensure correct automatic capitalization of text containing accented characters.

Message Translation

To simplify national language translation, the text of all messages and phrases used by the program has been gathered in DSMCSMDM (common messages), DSMCSPDB (common phrases), and DSMTSMDM (TSO messages). These modules are distributed in source form. In addition, the translated text of all messages and other literal strings used by the GML starter set are contained in macros DSM#Mxx and DSM#Sxx, where xx indicates a two number code. The following table illustrates the two-number code and the associated language:

Number	Language
01	American English
02	United Kingdom English
03	Dutch
04	German
05	Italian
06	French National
07	Spanish
08	Portuguese
09	Danish
10	Finnish
11	Norwegian
12	Swedish
13	Canadian French
14 – 16	Reserved for BookMaster
17	Canadian English
18	Icelandic

Table 8. Two-Number Code and It's Associated Language

Default Language

The default language of SCRIPT/VS can be changed to correspond to the language of your installation. This will eliminate the need for each user at your installation to specify the .DL [Dictionary List] control word to use your language's spelling and hyphenation dictionaries and hyphenation algorithms. The LANGUAGE attribute of the :GDOC tag uses the default language in its processing, so it is important for the default language to be correct for your installation. See "The DSMDEF Macro" on page 119 for instructions on changing the default language.

Default Font for Page Mode Devices

The default font used by SCRIPT/VS can be changed to correspond to the proper output code page for your installation. This will eliminate the need for each user at your installation to specify the CHARS option to use the correct default font. Using the correct font for page mode devices is necessary for accented characters to print correctly. See "The DSMLDT Macro" on page 131 for instruction on changing the default font. The following list shows the code pages that should be used for the various languages supported by SCRIPT/VS:

Code Page	Language
T1GI0386	Danish, Norwegian
T1GI0395	Dutch, English (U.S.), English (Canadian), French (Canadian), Icelandic, Portuguese
T1GI0394	English (U.K.)
T1GI0387	Finnish, Swedish
T1GI0385	French National
T1GI0382	German
T1GI0389	Italian
T1GI0303	Spanish

Katakana

Katakana characters and symbols can be processed by SCRIPT/VS. The following hexadecimal codes will not be used for syntactical purposes (operators or separators):

- Characters

58-5B
81-8A 8C-8F
90-9A 9D-9F
A2-AA AC-AF
BA-BF

- Symbols

41-49
51-56

Uppercase String Translation

SCRIPT/VS allows a single lowercase character to be translated to a string of uppercase characters. For example, the sharp s small “ß” character can be translated to “SS” with the .TU [Translate Uppercase] control word. Refer to the *DCF: SCRIPT/VS Language Reference* for more information about the .TU [Translate Uppercase] control word.

Other Considerations

No specific support for Kanji or Braille is included in DCF.

PostScript Code Pages

To modify PostScript code pages shipped with DCF Release 4.0, see “PostScript Code Pages” on page 279.

Chapter 13. Index Sort Sequence

Chapter 13. Index Sort Sequence	177
Sort Sequences	177
Modifying the Index Sort Sequences	181
The 1-to-2 Table	181
The 1-to-1 Table	181
Assigning Sort Values	182
Header Characters	183
Character Code Points	184
The Index Sort Modules	184

Chapter 13. Index Sort Sequence

SCRIPT/VS supports fifteen languages, each with different requirements for sorting entries in an index. Therefore, a set of language-specific translate tables (two for each of the fifteen IBM-supplied languages) is used to determine the sort sequence for the index entries.

The following list defines the basic sort sequence, which is common to all of the supported languages. SCRIPT/VS sorts the index entries in this order:

- Blank (code point 40)
- Required blank (code point 41) and characters to be "omitted." Omitted characters, such as the hyphen, syllable hyphen and the apostrophe, do not have unique sort values. They sort as if they were required blanks.
- Special Characters
- Numeric characters (sorted numerically)
- Alphabetic characters including the associated accented characters (sorted alphabetically).

In addition to the common sorting conventions listed above, the sort sequence for each language includes the specific sorting requirements of that language. The specific sort sequence used is determined by either the .DL INDEX control word or the default language if .DL INDEX is not specified. This allows the index sort sequence language to be different from the hyphenation and spelling verification language.

Sort Sequences

The following table lists the sort sequence used for each language. Characters that sort the same are grouped together within parentheses.

Table 9 (Page 1 of 3). Sort Sequences for the Languages Supported by SCRIPT/VS	
Language	Sort Sequence
Danish	(blank) (hyphen, apostrophe, required blank) special characters (sorted in order of code point values) 0 ¼ ½ ¾ 1 ' 2 ² 3 ³ 4 5 6 7 8 9 (a A á Â à Ã ä Å) (ä Å) (b B) (c C ç Ç) (d D ð Ð) (e E é Ê è Ë ê Ì ë Æ) (f F) (g G) (h H) (i Í Î Ï Ì Ï Î Ï) (j J) (k K) (l L) (m M) (n N ñ Ñ) (o O ó Ô õ Ö ö Ö Ö) (p P) (q Q) (r R) (s S) ß sorted as the character string "ss" (t T) (þ Þ sorts as "th") (u U ú Ú ù Û ü Ü) (v V) (w W) (x X) (y Y ý Ý ÿ Ü) (z Z) (æ Æ ä Ä) (æ and Æ sorted as the character strings "ae" and "AE") (ø Ø ö Ö) (å Å aa AA aA Aa)
Canadian French Dutch English American English Canadian English United Kingdom French National German Italian Portuguese	(blank) (hyphen, apostrophe, required blank) special characters (sorted in order of code point values) 0 ¼ ½ ¾ 1 ' 2 ² 3 ³ 4 5 6 7 8 9 (a A á Â à Ã ä Å) (ä Å ä Ä) æ and Æ sorted as the character strings "ae" and "AE" (b B) (c C ç Ç) (d D ð Ð) (e E é Ê è Ë ê Ì ë Æ) (f F) (g G) (h H) (i Í Î Ï Ì Ï Î Ï) (j J) (k K) (l L) (m M) (n N ñ Ñ) (o O ó Ô õ Ö ö Ö Ö) (ø Ø ö Ö) (p P) (q Q) (r R) (s S) ß sorted as the character string "ss" (t T) (þ Þ sorts as "th") (u U ú Ú ù Û ü Ü) (v V) (w W) (x X) (y Y ý Ý) (z Z)

Table 9 (Page 2 of 3). Sort Sequences for the Languages Supported by SCRIPT/VS

Language	Sort Sequence
Finnish Swedish	(blank) (hyphen, apostrophe, required blank) special characters (sorted in order of code point values) 0 ¼ ½ ¾ 1 ¹ ² ³ ⁴ ⁵ ⁶ ⁷ ⁸ ⁹ (a A á Ä à Å ä Ã) (ä Ä) (b B) (c C ç Ç) (d D ð Ð) (e E é Ê ë Ë ê Ì Ì) (f F) (g G) (h H) (i I í Î ï Ï ï Ì) (j J) (k K) (l L) (m M) (n N ñ Ñ) (o O ó Ô õ Ö ö Ö) (ö Ö) (p P) (q Q) (r R) (s S) ß sorted as the character string "ss" (t T) (þ Þ sorts as "th") (u U ú Ú ù Û ü Ü) (v V) (w W) (x X) (y Y ý Ý ÿ ü Ü) (z Z) (ä Ä) (ä Ä æ Æ) (ö Ö ø Ø)
Icelandic	(blank) (hyphen, apostrophe, required blank) special characters (sorted in order of code point values) 0 ¼ ½ ¾ 1 ¹ ² ³ ⁴ ⁵ ⁶ ⁷ ⁸ ⁹ (a A à Á â Ã) (ä Ä å Å ä Å) (á Á) (b B) (c C ç Ç) (d D ð Ð) (e E è Ê ë Ë ê Ì Ì) (é É) (f F) (g G) (h H) (i I í Î ï Ï ï Ì) (í Í) (j J) (k K) (l L) (m M) (n N ñ Ñ) (o O ò Ô õ Ö ö Ö) (ó Ó) (p P) (q Q) (r R) (s S) ß sorts as the character string "ss" (t T) (u U ù Û ü Ü) (ú Ú) (v V) (w W) (x X) (y Y ý) (ý Ý) (z Z) (þ Þ) (æ and Æ) (ö Ö ø Ø)

Table 9 (Page 3 of 3). Sort Sequences for the Languages Supported by SCRIPT/VS

Language	Sort Sequence
Norwegian	(blank) (hyphen, apostrophe, required blank) special characters (sorted in order of code point values) 0 ¼ ½ ¾ 1 ¹ ² ³ ⁴ ⁵ ⁶ ⁷ ⁸ ⁹ (a A á Ä à ã Ã) (ä Å ä Ä) (b B) (c C ç Ç) (d D ð Ð) (e E é Ê ë Ë ê ì ï) (f F) (g G) (h H) (i I í î ï Ì Î Ï) (j J) (k K) (l L) (m M) (n N ñ Ñ) (o O ó Ó ò Õ õ Ö) (ö Ö ö Ö) (p P) (q Q) (r R) (s S) ß sorted as the character string "ss" (t T) (þ Þ sorts as "th") (u U ú Ú ù Û ü Ü ù Ü) (v V) (w W) (x X) (y Y ý Ý ÿ) (z Z) (æ Æ) (ø Ø) (å Å aa AA aA Aa)
Spanish	(blank) (hyphen, apostrophe, required blank) special characters (sorted in order of code point values) 0 ¼ ½ ¾ 1 ¹ ² ³ ⁴ ⁵ ⁶ ⁷ ⁸ ⁹ (a A á Ä à ã Ã) (ä Å ä Ä) æ and Æ sorted as the character strings "ae" and "AE" (b B) (c C ç Ç) (ch cH Ch CH) [sort as one character] (d D ð Ð) (e E é Ê ë Ë ê ì ï) (f F) (g G) (h H) (i I í î ï Ì Î Ï) (j J) (k K) (l L) (ll ll Ll Ll) [sort as one character] (m M) (n N ñ Ñ) (o O ó Ó ò Õ õ Ö) (ø Ø ö Ö ö Ö) (p P) (q Q) (r R) (s S) ß sorted as the character string "ss" (t T) (þ Þ sorts as "th") (u U ú Ú ù Û ü Ü ù Ü) (v V) (w W) (x X) (y Y ý Ý ÿ) (z Z)

Modifying the Index Sort Sequences

The translate tables can be modified if your installation requires a sort sequence other than the ones defined above. This section explains how SCRIPT/VS uses these tables and what you need to be aware of when you modify these tables. The translate tables are defined in a SCRIPT/VS module called DSMTRNLT. “DSMTRNLT: National Language Character Definitions” on page 144 describes how you can modify this module at installation.

Each index entry sort key is a string of characters. This string is represented internally by a string of code point values. Because the code point values assigned to the characters do not represent the desired sort order, these values are mapped to sort values. The index entry sort keys are translated to these sort values using a two-step process. The first step uses a translation table called the 1-to-2 table. The second step uses a translation table called the 1-to-1 table. There are 18 translation tables and each language uses 2 tables. Some of the translation tables are used by more than one language.

The 1-to-2 Table

The 1-to-2 tables are defined in the DSMTRNLT module, using DSMIXTT2 macro calls. The 1-to-2 table is used to translate certain single characters to a 1- or 2-character string that should be used when sorting. For example, the “Æ” diphthong character should be sorted as the character string “AE.” Therefore, the 1-to-2 table assigns “Æ” = “AE.” Another example is the “ß” character, which should be sorted as the character string “ss.” These translations are examples of 1-character to 2-character translations. Other translations in this table are 1-character to 1-character translations. For example, the single characters “a,” “ä,” and “Â” are all translated to “A,” because they should all sort the same as “A.” The unaccented, uppercase form of a character is called its header character. Header characters are discussed later in this section.

The following is an example of translating an index entry sort key using the 1-to-2 table. Given the key:

straße

the 1-to-2 table is used to translate the “ß” to “ss” and to translate the other characters to their header characters:

STRASSE

Default: Each 512 byte 1-to-2 table is initialized to zeros. Zero indicates that no translation needs to be done for a character. This means that any character in a key that is not assigned a translation value in the 1-to-2 table will remain unchanged.

The 1-to-1 Table

The 1-to-1 tables are defined in the DSMTRNLT module using DSMIXTT1 macro calls. The 1-to-1 table is used to translate a character to its sort value. Following is an example of some sort values that could be assigned using a 1-to-1 table.

A = 01	Ä = 02	B = 03	C = 04	D = 05	E = 06	F = 07	G = 08	H = 09
I = 0A	J = 0B	K = 0C	L = 0D	M = 0E	N = 0F	O = 10	P = 11	Q = 12
R = 13	S = 14	T = 15	U = 16	V = 17	W = 18	X = 19	Y = 1A	Z = 1B

Note: The sort values are assigned in hexadecimal.

Sort values must be assigned, because the code point values of the characters do not necessarily represent the order in which the characters should be sorted. For example, the code point value of "Ä" may be 4A and the code point value of "A" may be C1. If the code point values were used to sort the character, the character "Ä" would sort before the character "A" with all the characters from 4B to C0 in between. In the example above, "Ä" should sort after the "A" so "A" is assigned hexadecimal value 01 and "Ä" is assigned hexadecimal value 02.

The following example uses the sample 1-to-1 table above to show how the two-step translation process works. Once again using the key:

straße

the 1-to-2 table is used to translate the key to:

S	T	R	A	S	S	E
---	---	---	---	---	---	---

Then the 1-to-1 table is used to translate the characters to their sort values. The final result is:

14	15	13	01	14	14	06
----	----	----	----	----	----	----

Default: Each 256 byte 1-to-1 table is initialized to identity, that is, each character initially maps to itself. Any character in a key that is not assigned a translation value in the 1-to-1 table will remain unchanged. If no characters are assigned sort values in the 1-to-1 table, the result is a sort sequence based on the code point values for the characters. This is sometimes called a hexadecimal sort.

Assigning Sort Values

The sort value assigned to a character can be any hexadecimal value from 00 to FF (255 decimal). When you assign sort values, the numbers do not have to be contiguous. For example:

A = 01, B = 03, C = 05, D = 07, ...

would produce the same index output as if the following values had been used:

A = 01, B = 02, C = 03, D = 04, ...

Numbers can be skipped to leave room for accented characters. For example:

A = 01, Ä = 02, B = 03, C = 04, Ç = 05, D = 06, Ð = 07, E = 08, ...

In the previous examples we always started with the number 1, but there is no reason that we have to start with that or any other number. If the numbering started at 40, the results would be the same, and there would be room to put other characters before the "A," if desired:

& = 30, % = 31, A = 40, Ä = 41, B = 42, C = 43, ...

Header Characters

The header character is the character printed at the top of each group of index entries. For example, words starting with the characters "a," "A," "á," "Á," "â," "Â" appear as one group in the index under the header "A."

The 1-to-2 table is used to translate characters in a key to their header characters. Then, the first character in the key is saved as the header character before all of the characters are translated to their sort values using the 1-to-1 table. The header character for each group of index entries comes from the first entry in that group. The example below shows how this works. Even though the first entry starts with a lowercase "a," the "a" in the key is translated to an uppercase "A," and that character is used for the header:

A

a word
A word
á word
Á word
â word
Â word

It is important to translate every character to its header character using the 1-to-2 table. If this is not done, the header for a group of entries may not be correct. For example, if the first entry started with a lowercase "a," as in the example below, and the "a" in the key was not translated to an uppercase "A," then the header printed above that group of entries would be a lowercase "a" instead of an uppercase "A":

a

a word
A word
á word
Á word
â word
Â word

The same type of problem would occur if the key for the first entry started with an "Ä." The "Ä" would be printed as the header instead of an uppercase "A."

Character Code Points

The sort-sequence translation tables set up character-to-character and character-to-sort value mappings. Characters that are not on the keyboard cannot be entered into the table without using their code point values. Any character can be referred to using the code point value for that character. For example, code point "C1" can be used for "A." The code point values for the characters depend on the code page being used. Following is a list of the code pages used for the IBM-supplied sort sequences. The sort-sequence tables for your installation may have to be modified if the code pages you use do not match these code pages.

Danish	Code Page 00277
Dutch	Code Page 00037
English American	Code Page 00037
English Canadian	Code Page 00037
English United Kingdom	Code Page 00285
Finnish	Code Page 00278
Canadian French	Code Page 00037
French National	Code Page 00297
German	Code Page 00273
Icelandic	Code Page 00871
Italian	Code Page 00280
Norwegian	Code Page 00277
Portuguese	Code Page 00282
Spanish	Code Page 00284
Swedish	Code Page 00278

These are Version 1 of the Country Extended Code Pages (CECP). These code pages can be found in the *ABOUT TYPE: IBM's Technical Reference* for the fonts you plan to use.

The Index Sort Modules

There are fifteen index sort modules, which correspond to the fifteen supported languages. These modules are named DSMxxIXS, where xx is the first two letters of the abbreviated form of the corresponding language. For example, DSMEAIXS is English American (EAM). When a .PI control word is encountered, the key is passed to the appropriate DSMxxIXS modules, which then translate the key. The module used is determined by the sort-sequence language for the document, which can be specified using .DL INDEX if the default language is not the desired language.

These modules translate the key, using the 1-to-1 and 1-to-2 tables, and handle the special requirements of each language. The interface to these modules is documented below so that you can replace them if special processing is needed.

The input to DSMxxIXS is as follows:

R1 contains the address of a parameter list which contains:

- The address of the buffer holding the key
- The length of the key
- The address of the place where the header string should be returned

- The address of the DSMxxXT1 and DSMxxXT2 tables
- The address of a temporary buffer.

DSMxxIXS translates index entry sort keys as follows:

1. Translates certain single characters to their 1- or 2-character representations and translates characters to their header characters using the 1-to-2 table. The 1-to-2 table name is DSMxxXT2, where xx matches the xx in the module name. Some characters will not need to be translated. If this is the case, the character translation value in the 1-to-2 table will be zero, and the character will not be replaced with another character. If the character translation value is only one character instead of two, the second character position in the table will be zero.
2. Saves the first character in the translated key to be returned for the header string. When the index is processed, a header is printed at the top of each group of entries that sort together in the index.
3. Translates all of the characters to their sort values using the 1-to-1 table. The 1-to-1 table name is DSMxxXT1, where xx matches the xx in the module name.
4. The DSMSPIXS module for Spanish has two additional steps:
 - a. Replaces the character strings "ch," "Ch," and "CH" with a sort value that causes these character strings to sort after the character "Ç" and before the character "D." If one of these strings is found at the beginning of the key, "CH" is saved as the header string.
 - b. Replaces the character strings "ll," "ll," "ll," "ll," and "ll" with a sort value that causes these character strings to sort after the character "L" and before the character "M." If one of these strings is found at the beginning of the key, "ll" is saved as the header string.

Note: Hexadecimal code point B3 found between two Ls is interpreted as a middle dot.

5. The DSMDAIXS and DSMNOIXS modules for Danish and Norwegian have one additional step:

Replaces the character strings "aa," "Aa," "aA," and "AA" with a sort value that causes these character strings to sort after the character Å.

The output of the DSMxxIXS module is as follows:

- The length of the new index sort key stored at the address passed in the parameter list
- The translated key which replaced the input key
- The header string stored at the address passed in the parameter list.

Refer to "Procedure for Tailoring SCRIPT/VS" on page 111 for information about how to incorporate a module into SCRIPT/VS.

Chapter 14. Using Separation Masters

Chapter 14. Using Separation Masters	189
Creating Separation Masters	189
Steps to Creating Separation Masters	191
Using the SEPMASTR Command Option	191
Using the .SM Control Word	192
Coding for the Sample Form	192
Using SEPMASTR Keyword with Control Words	192
.DF Control Word	192
Coding for the Sample Form	193
.DR Control Word	193
Coding for the Sample Form	193
.OI Control Word	194
.PO Control Word	194
.SD Control Word	194
Coding for the Sample Form	195
.SI Control Word	195
Coding for the Sample Form	195
Default Master	195
Producing Separation Masters	196
Creating Multiple Color Graphics	200
Modifying the GML Profiles	201
Fonts	201
Rules and Tables	202

Chapter 14. Using Separation Masters

Separation masters are used in the printing process to produce multiple-color documents. When masters are sent to a printing vendor, you need to mark which graphics, rules, shades, or text you want printed in a color other than black. This can be done in different ways. For example, you can make a *printer's dummy*, which is a copy of the masters with items marked to indicate what should be printed in a color other than black. This works well if the final printing uses just one color other than black. If it will be printed with more than one other color, then overlays (one for each color) need to be created. With the separation masters function, SCRIPT/VS will create a separate master for each color.

The separation master function allows you to produce separate masters by formatting the document only once. Each master can then be used either as part of a multiple-part form or with the offset printing process to produce color separation masters. You can put graphics, rules, shades, and text in different masters by using the control words with the SEPMASTR keyword. These control words are listed in "Steps to Creating Separation Masters" on page 191.

Note: All masters are printed in black.

Creating Separation Masters

The following figure shows an insurance form with document elements that will be coded for separate masters. The document elements used are: fonts, tables, shading, and page segments. The coding used to create the form is shown in "Producing Separation Masters" on page 196 and a reduced copy of the form is shown in Figure 3 on page 190.



Plan Number:
Branch:
Plan Member ID:
Claim Number:
Date Received:

Plan Member:
Patient Name:
Relationship:
Date Processed:
Micro Film Number:

[illegible]

INSURANCE SPECIALISTS
P.O. Box 1000
Boulder, Colorado 80301



Figure 3. Sample Insurance Form Used with Separation Masters. The page shown in this example is a reduced copy of the printed page.

Steps to Creating Separation Masters

There are three main steps to creating separation masters with SCRIPT/VS:

- Use the SEPMASTR command option to specify which numbered masters (assigned with the .SM control word) you want to produce.
- Use the .SM [Separation Masters] control word to assign the *named* masters to *numbered* masters.
- Use the following control words with the SEPMASTR keyword to associate those document elements with a *named* master:
 - .DF [Define Font]
 - .DR [Define Rule]
 - .OI [Overlay Include]
 - .PO [PostScript]
 - .SD [Shade Definition]
 - .SI [Segment Include].

Note: This function is ignored for non-page devices.

Using the SEPMASTR Command Option

Use the SEPMASTR command option to select which masters you want to produce. For example, to produce masters numbered 4, 5, and 8, enter

```
SCRIPT SEPMASTR(4 5 8)
```

If you specify SEPMASTR with no suboptions, then all the separation masters defined with the .SM control word will be produced, including the default master.

SCRIPT/VS issues a message for each logical output page indicating the page number, the values of the separation masters selected for that page, and the order they were produced. If you use MESSAGE(DELAY) with the SEPMASTR command option, the messages will be saved in the default master when the document is formatted.

If the SEPMASTR command option is not specified, the document items defined with the SEPMASTR keyword and the .SM [Separation Masters] control word will be ignored.

The &\$SEPM system symbol indicates which SEPMASTR command options were specified by returning a string of 1s and 0s (separated by blanks). For example, if you entered the command option SEPMASTR(4 5 8) and then queried the symbol &\$SEPM, SCRIPT/VS would return:

```
0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0
```

The suboptions are numbers that correspond to the first parameter on the .SM control word.

Using the .SM Control Word

The .SM [Separation Masters] control word groups document elements into separate masters that can be used to produce multi-part forms or with the offset printing process to produce multi-color documents. Use the .SM control word to define which document elements will go into which named master. The name used to identify the document elements is the name used with the SEPMASTR keyword. In the following example, the document elements that are changed in the profile examples in "Modifying the GML Profiles" on page 201 are assigned to numbered separation masters.

```
.sm 1 head1
.sm 2 xmpls
.sm 3 frame
```

Head level 1s will be placed in separation master 1. Examples will be placed in separation master 2. Rules that frame tables will be placed in separation master 3.

Coding for the Sample Form

The following coding was used to define the document elements in the numbered separation masters in Figure 4 on page 199.

```
.sm 1 thckrule thinrule tbshade
.sm 2 titlmas
.sm 3 segmas
```

The rules drawn with the THICK rule and BOXRULE rules and anything shaded with the named shade SEPSHADE will be placed in separation master 1. The text created with the TITLE font will be placed in separation master 2. The page segments assigned to SEGMAS will be placed in separation master 3. The default master will contain all other document elements.

Using SEPMASTR Keyword with Control Words

Use the following control words with the SEPMASTR keyword to associate those document elements with a named master:

- .DF [Define Font]
- .DR [Define Rule]
- .OI [Overlay Include]
- .PO [PostScript]
- .SD [Shade Definition]
- .SI [Segment Include].

.DF Control Word

Use the SEPMASTR keyword with the .DF [Define Font] control word to specify that text printed in a particular named font be assigned to a named master. For example:

```
.df first font x0a075di sepmastr frstmast
.df second type('sonoran sans serif' 12 bold) sepmastr scndmast
```

Text printed in the FIRST font (X0A075DI, which is 14-point Sonoran Sans Serif bold) will be assigned to a named master called FRSTMAST. Text assigned to

the SECOND font (12-point Sonoran Sans Serif bold) will be assigned to a named master called SCNDMAST.

Assign those named masters to numbered masters with the .SM [Separation Masters] control word.

```
.sm 1 frstmast
.sm 2 scndmast
```

When the SEPMASTR(1 2) option is used with the SCRIPT command, text in each named font will be placed in the appropriate master.

Coding for the Sample Form

The following coding was used to define the fonts used in Figure 4 on page 199.

```
.df title type('sonoran sans serif' 12 bold) sepmastr titlmas
.df table type('sonoran sans serif' 8 bold)
.df tbod type('sonoran sans serif' 8)
.df body type('sonoran sans serif' 10)
.df addr type('sonoran sans serif' 9)
```

Only the font named TITLE was defined to a master named TITLMAS. Because the other fonts are not assigned to an explicit master, they will go into the default master.

.DR Control Word

Use the SEPMASTR keyword with the .DR [Define Rule] control word to specify that certain named rules be assigned to a named master. For example:

```
.dr firstblu weight 1mm sepmastr blurule
.dr secongrn weight 2mm sepmastr grenrule
```

When FIRSTBLU and SECONGRN named rules are drawn, those rules will be assigned to the named master specified with the .DR control word.

Assign those named masters to numbered masters with the .SM [Separation Masters] control word.

```
.sm 3 blurule
.sm 4 grenrule
.sm 7 blurule
```

When the SEPMASTR(3 4 7) option is used with the SCRIPT command, those rules will print on separate pages. Note that the rule named FIRSTBLU, which was assigned to a named master called BLURULE, will print twice, once in master 3 and once in master 7.

Coding for the Sample Form

The following coding was used to define the rules used in Figure 4 on page 199.

```
.dr thick weight p2 sepmastr thckrule
.dr boxrule sepmastr thinrule
```

The rule named THICK was defined to a master named THCKRULE. The rule named BOXRULE was defined to a master named THINRULE. Because BOXRULE is the default rule used with tables, all tables using the default table rules will be assigned to the THINRULE named master also. Any other rules will go into the default master.

.OI Control Word

Use the SEPMASTR keyword with the .OI [Overlay Include] control word to specify that an overlay be included in an output master other than the default master. For example:

```
.oi example hpos 0 vpos 0 sepmastr mastr5
```

The overlay named EXAMPLE will be assigned to a named master called MASTR5.

Assign the named master to a numbered master with the .SM [Separation Masters] control word.

```
.sm 5 mastr5
```

When the SEPMASTR(5) option is used with the SCRIPT command, the overlay will be placed in separation master 5.

.PO Control Word

Use the SEPMASTR keyword with the .PO [PostScript] control word to specify that a PostScript image should be included in an output master other than the default master. For example:

```
.po circle sepmastr blupost
```

The PostScript image named CIRCLE will be assigned to a named master called BLUPOST.

Assign the named master to a numbered master with the .SM [Separation Masters] control word.

```
.sm 6 blupost
```

When the SEPMASTR(6) option is used with the SCRIPT command, the PostScript image will be placed in separation master 6.

.SD Control Word

Use the SEPMASTR keyword with the .SD [Shade Definition] control word to specify that a named shade should be included in an output master other than the default master. For example:

```
.sd sdarea standard shade light sepmastr grnshade
```

The shade named SDAREA will be assigned to a named master called GRNSHADE.

Assign the named master to a numbered master with the .SM [Separation Masters] control word.

```
.sm 7 grnshade
```

When the SEPMASTR(7) option is used with the SCRIPT command, each time the shade named SDAREA is to print, the shading will be placed in separation master 7.

Coding for the Sample Form

The following coding was used to define the shade used in Figure 4 on page 199.

```
.sd sepshade shade 32 sepmastr tbshade
```

The shade named "sepshade" was defined to a master named "tbshade."

.SI Control Word

Use the SEPMASTR keyword with the .SI [Segment Include] control word to specify that a page segment be included in an output master other than the default master. For example:

```
.si seg1 sepmastr bluseg  
.si seg2 sepmastr grnseg
```

The page segment named SEG1 will be assigned to a named master called BLUSEG. The page segment named SEG2 will be assigned to a named master called GRNSEG.

Assign the named masters to numbered masters with the .SM [Separation Masters] control word.

```
.sm 8 bluseg  
.sm 9 grnseg
```

When the SEPMASTR(8 9) option is used with the SCRIPT command, SEG1 will be placed in separation master 8 and SEG2 will be placed in separation master 9.

Coding for the Sample Form

The following coding was used to define the page segments used in Figure 4 on page 199.

```
.si logois inline sepmastr segmas  
.si regism inline sepmastr all
```

The page segment named "logois" was defined to a master named "segmas." The page segment named "regism" is the registration mark. Because SEPMASTR ALL was specified, the registration mark will print on all masters. (This is what the printing vendor will use to ensure that the separate masters are lined up correctly.)

Default Master

The default master consists of every document element that is not identified with a SEPMASTR keyword on the .DF, .DR, .OI, .PO, .SD, or .SI control words. You can control or add to the contents of the default master with the .SM control word with a first parameter of zero (.SM 0).

If you are not requesting all the separation masters, you have to explicitly specify SEPM(0) to get the default master.

Producing Separation Masters

Once the document elements are defined to separate named masters, and those named masters are assigned to numbered masters you can create your document. The following example shows most of coding used to create the sample form in Figure 3 on page 190. Much of the coding used to build the table was omitted because it was repetitive.

The recommended method of coding separation masters is to place the separation definitions at the beginning of the document. This ensures that the named masters have been associated with a numbered master, thereby producing the correct separations when the document is printed.

```
.*****
.ti ~ 05
.df title type('sonoran sans serif' 12 bold) sepmastr titlmas
.df table type('sonoran sans serif' 8 bold)
.df tbod type('sonoran sans serif' 8)
.df body type('sonoran sans serif' 10)
.df addr type('sonoran sans serif' 9)
.dr thick weight p2 sepmastr thckrule
.dr boxrule sepmastr thinrule
.da logo1 42p6 4p width .5i
.da logo2 6p6 39p width .5i
.da reg1 2p 2p
.da reg2 46p 62p
.sd sepshade shade 32 sepmastr tbshade
.sm 1 thckrule thinrule tbshade
.sm 2 titlmas
.sm 3 segmas
.td table medplan hrules thick vrules thick
.td row head arrange (1 2 2 3 4 4 4 / 1 5 6 3 7 8 9)
.td row head cwidths (9p 4p6 4p6 5p 5p 3p 5p6) hrules thick
.td cell 1 head center
.td cell 2 head
.td cell 3 head center
.td cell 4 head
.td cell 5 head center
.td cell 6 head center
.td cell 7 head
.td cell 8 head center
.td cell 9 head
.td row benesd cwidths (9p 4p6 4p6 5p 5p 3p 5p6) hrules none
.td cell 1 benesd shade sepshade
.td cell 2 benesd shade sepshade
.td cell 3 benesd shade sepshade
.td cell 4 benesd shade sepshade
.td cell 5 benesd shade sepshade
.td cell 6 benesd shade sepshade
.td cell 7 benesd shade sepshade
.td row noshad cwidths (9p 4p6 4p6 5p 5p 3p 5p6) hrules none
.td cell 1 noshad
.td cell 2 noshad
.td cell 3 noshad
.td cell 4 noshad
.td cell 5 noshad
.td cell 6 noshad
.td cell 7 noshad
.td row total cwidths (18p 5p 5p 3p 5p6)
.td cell 1 total
.td cell 2 total
.td cell 3 total
.td cell 4 total shade sepshade
.td cell 5 total
.td row comment cwidths (31p 5p6)
.td cell 1 comment center
.td cell 2 comment
.*****
.ar logo1 on
.si logois inline sepmastr segmas
.ar logo1 off
```

```

.ar reg1 on
.si regism inline sepmastr all
.ar reg1 off
.ar reg2 on
.si regism inline sepmastr all
.ar reg2 off
.nf center
.bf title
IBM SURGICAL PLAN
.bf body
Claim Administration Provided By
The Insurance Specialists Insurance Company
P.O. Box 1000
Boulder, Colorado 80301

.bf title
EXPLANATION OF BENEFITS
.sp 3p
.tp 18p
.nf left
.bf tbod
Plan Number:      ~Plan Member:
Branch:           ~Patient Name:
Plan Member ID:   ~Relationship:
Claim Number:     ~Date Processed:
Date Received:    ~Micro Film Number:
.nf center
.sp
.bf table
.ta table medplan on
.ta row head on
.ta cell 1 on
Name of Provider
.ta cell 2 on
Dates of Service
.ta cell 3 on
Charge
.ta cell 4 on
Benefit Calculation
.ta cell 5 on
From
.ta cell 6 on
To
.ta cell 7 on
Eligible
Amount
.ta cell 8 on
%
.ta cell 9 on
Benefit
Amount
.ta row benesd on
.ta cell 1 on
.sp 1
.ta row noshad on
.ta cell 1 on
.sp 1
.ta row benesd on
.ta cell 1 on
.sp 1
.ta row noshad on
.ta cell 1 on
.sp 1
.ta row total on
.in p6
.ta cell 1 on
.nf left
Column Totals
.ta row comment on
.ta cell 1 on
* See Claim Explanation Area on Reverse Side

```

```

.ta cell 2 on
.ta row comment on
.ta cell 1 on
.ir p6
.sp 2
.nf right
Net Payment
.ir
.ta table off
.sp 2
.fo left
.ar logo2 on
.si logois inline sepmastr segmas
.ar logo2 off

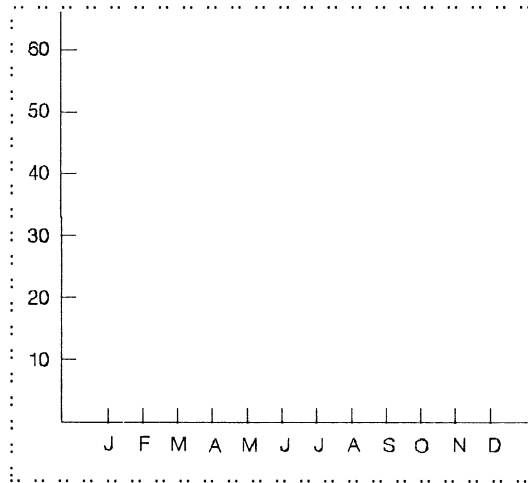
.nf left
.tp 5p
.bf title
~INSURANCE SPECIALISTS
.bf addr
~P.O. Box 1000
~Boulder, Colorado 80301
.nf off

```

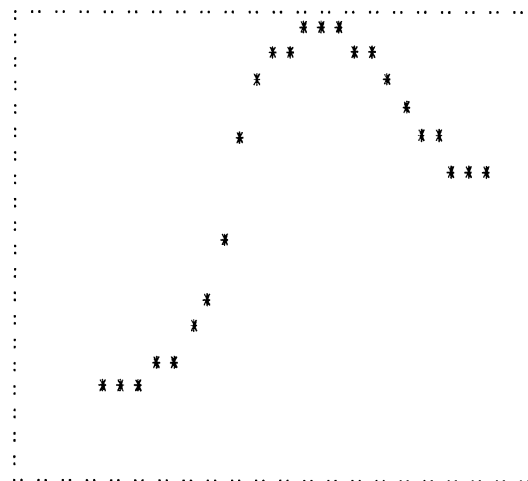
When the document is formatted with SCRIPT/VS and SEPMASTR(0 1 2 3) is used, you will get output as shown in Figure 4 on page 199.

Creating Multiple Color Graphics

It is possible to have graphics print with multiple colors by creating one page segment for each part of the graphic you want a different color and then overlapping them when they are printed. For example, suppose the CMS file AXES PSEG3820 contains the following image, which you want printed black.



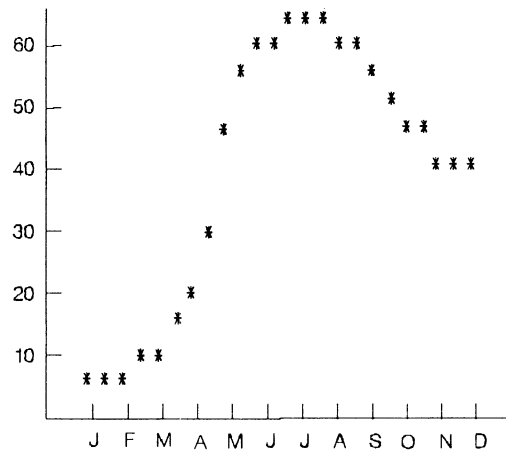
In addition, the CMS file PLOT1 PSEG3820 contains the following image, which you want printed blue:



These segments can be coded so that they will overlap each other when printed, as follows:

```
.si axes
.ls normal 0
.si plot1 sepmastr bluplot
.ls
```

When the document is sent to a printing vendor and the masters are combined in the printing process, they will print as follows:



This same technique could be used with overlays using .OI [Overlay Include] or with PostScript images using .PO [PostScript].

Modifying the GML Profiles

To assign document elements to specific masters without changing the document source, modify the GML profiles. This will ensure consistency throughout your library and allow you to use the separation master function with existing documents without changing them.

Profile Name	Application
DSMPROF4	GML Starter Set
DSMTPROF	Overhead transparency tags
DSMMPROF	Memo tags
DSMSPROF	Schedule tags

After you update the profile, use the .SM [Separation Masters] control word to assign the document elements to a numbered separation master. See “Using the .SM Control Word” on page 192 for more information on the .SM [Separation Masters] control word. To ensure consistency, the .SM control word should go in the profile also.

Fonts

If you were preparing a set of manuals to send to a printing vendor and you wanted all the head-level 1s to be printed in blue and all the examples to be printed in green, you could modify the starter set to have SCRIPT/VS automatically create separate masters for each page that you want the printing vendor to print with a color other than black.

Search through the profile file until you locate the font definitions for page devices. Assign the hd1 and xmpfont definitions to a named master with the SEPMASTR keyword as shown in Figure 5 on page 202.

```

.sm 1 head1
.sm 2 xmpls

:
.go endpgfn
...PG2
.*df figfont
.df xmpfont type(prestige 9) codepage t1d0base sepmastr xmpls
.df @rh type(bold 10)
.df @rf type(bold 10)
.df ieh type(bold 10)
.df hi0 type(normal)
.df hi1 type(italic)
.df hi2 type(bold)
.df hi3 type(bold) up
.df super type(8)
.df fnt type(10)
.df title type(bold 24) up
.df author type(12)
.df address type(10)
.df docnum type(italic 10)
.df date type(italic 10)
.df titlesec type(bold 10)
.df hd0 type(bold 16) up
.df hd1 type(bold 16) up sepmastr head1
.df hd2 type(bold 16)
.df hd3 type(bold 10) up
.df hd4 type(bold 10)
.df hd5 type(bold)
.df hd6 type(italic)
.df hd0toc type(bold 10) up
.df hd1toc type(bold 10)
.df hd2toc type(10)
.df hd3toc type(10)
.df figcap type(bold 10)
.df figdesc type(10)
.df lqfont type(10)
.go endpgfn

:

```

Figure 5. Example of Modifying Part of a GML Profile

Rules and Tables

If you were preparing a set of manuals to send to a printing vendor and you wanted all the rules around the tables to print in red, you could modify the starter set to have SCRIPT/VS automatically create separate masters for each page that has tables with a box frame.

Search through the profile file until you locate the rule definitions. Assign the thick definition to a named master with the SEPMASTR keyword as shown in Figure 6 on page 203.


```

.sm 3 frame

:
.
*****
*   Define some rules for use in the Starter Set   *
*****
.dr @figrule
.dr @fnldr w .2mm
.dr @t@thick weight .5mm sepmast frame
*****
*   Set up the defaults for tables.               *
*****
.gs args @t@thick @t@thick boxrule boxrule boxrule
.gs vars @thrules @tvrules @rhrules @chrules @cvrules
.se @t@state = ""
.td row @tnote hrules &@rhrules

:

```

Figure 6. Example of Assigning the Outside Table Rule to a Separate Master

Appendix A. Using SCRIPT/VS with Other IBM Programs

Appendix A. Using SCRIPT/VS with Other IBM Programs	207
Producing Input for STAIRS/VS	207
Specifying STAIRS/VS Output	207
Restrictions Imposed on Formatted Output	207
STAIRS/VS Paragraph Identification	208
Using SCRIPT/VS as a Postprocessor	209
Using SCRIPT/VS as a Preprocessor	210
Developing Preprocessor APFs and Profiles	210
Redefining Symbols	211
Handling Directly Entered Control Words	211
Managing a Source Document	211
Preparing for Processing	212

Appendix A. Using SCRIPT/VS with Other IBM Programs

You can use SCRIPT/VS to format an input stream prepared by another program. You can also use SCRIPT/VS as a preprocessor to prepare an input file for processing by another text-processing system or by an application program.

This appendix is intended to help the text programmer format an input stream prepared by another program, and prepare an input stream for processing by another program. It contains information pertaining to the Storage and Information Retrieval System/Virtual Storage (STAIRS/VS), using SCRIPT/VS as both a postprocessor and a preprocessor, and developing preprocessor APFs and profiles. This appendix documents Product Implementation Information provided by DCF.

Product Implementation Information is detailed design and implementation information that is provided to assist the customers in performing such tasks as modifying, tuning, and diagnosing which by their nature require the use of information that depends on the detailed design and implementation of DCF. This appendix documents no programming interfaces for use by customers in writing programs that request or receive the services of DCF.

Producing Input for STAIRS/VS

The Storage and Information Retrieval System/Virtual Storage (STAIRS/VS) is an IBM program product that provides content-based retrieval of documents using a comprehensive indexing structure. Documents to be stored in the STAIRS/VS data base must be prepared in a Condensed Text Format (CTF). SCRIPT/VS can provide input for STAIRS/VS in this format, as shown in Table 10 on page 209.

Specifying STAIRS/VS Output

Use the DEVICE option of the SCRIPT command to specify STAIRS/VS output. (For details, refer to the description of the DEVICE command option in the *DCF: SCRIPT/VS Language Reference*.)

When you specify DEVICE(STAIRS) or CTF, SCRIPT/VS formats the input document as it would for device 1403W6. The formatted lines are then converted to STAIRS/VS CTF blocks. These blocks are *fixed* length records of 1008 bytes, and they must be directed to either an F (fixed-length records) or FB (fixed-length, blocked records) type of data set. The use of a V (format-V records) or VB (variable-length, blocked records) data set may cause a system error.

If you specify DEVICE(STAIRS) and PRINT, FILE, or TERM, the document is written to the specified destination for proofreading.

Restrictions Imposed on Formatted Output

The STAIRS/VS program recognizes only the text in the body of a document. Consequently, running headings and footings, footnotes, and floats are ignored when preparing STAIRS/VS output. Multiple-column sections are treated as a single column. Underscore, overstrike, skip, and space are also ignored.

Whenever a paragraph has a paragraph identifier that is equal to or lower than the previous paragraph identifier, STAIRS/VS will create a new logical document. This is referred to as *stepdown*. This may or may not be the result of a user error. In either case, SCRIPT/VS will flag it as an error in *proofing* mode output as described below.

In addition, STAIRS/VS requires that its input not exceed 69 characters per line, or 449 lines per paragraph. In documents prepared for STAIRS/VS, lines exceeding these limitations are flagged when output is being prepared in *proof* format, and are truncated when output is being prepared in CTF records. The following error flags are placed in columns 72 through 80 of proof output:

C - more than 69 characters
 L - more than 449 lines
 P - paragraph id stepdown

A line is considered to be a sentence delimited by *full stops*: a full-stop character followed by two blanks. As described in the chapter titled "Composing Lines" in the *DCF: SCRIPT/VS User's Guide*, SCRIPT/VS will automatically insert an extra blank between sentences when an input line ends in a full-stop character, to satisfy this requirement.

STAIRS/VS Paragraph Identification

STAIRS/VS requires that a 3-character identifier be associated with each paragraph of a document placed in its data base.

STAIRS/VS paragraph identifiers are composed of a single decimal digit (0 through 9) followed by one or two alphameric characters in ascending order (blank, A through Z, and 0 through 9). SCRIPT/VS, by default, numbers the first paragraph of a document 0, and increments the paragraph number by one with each break when concatenation is on, or by one with each input line when concatenation is off.

Thus, the first and subsequent paragraphs will be numbered:

0	0AE	0A2	0BB
0A	0AF	0A3	0BC
0AA
0AB	0AZ	0A9	
0AC	0A0	0B	
0AD	0A1	0BA	

You can reset the STAIRS/VS paragraph-numbering counter at any time with the .SO [STAIRS/VS Output] control word. For example, specifying:

.so pid 20b

causes subsequent paragraphs to be numbered:

20B	20Z	208
20C	200	209
20D	201	21
...	...	21A
20X	206	21B
20Y	207	...

When the STAIRS/VS paragraph-numbering counter is reset to a value that is equal to or less than the last value used, a new logical document is created,

regardless of whether a new document name has been specified with the .SO DOC control word.

When output is prepared in condensed text format, the STAIRS/VS paragraph number is included as part of the CTF record with each line. When output is prepared in proof format, the STAIRS/VS paragraph number is printed to the left of the first line of each paragraph.

Information for the document name, operator number, and read and delete password fields of the CTF block can also be provided with the .SO [STAIRS/VS Output] control word.

Table 10. STAIRS/VS Condensed Text Format (CTF) Records. Each record has a fixed length of 1008 bytes and contains up to 13 lines of text.			
Offset		Length	Contents
Dec.	Hex		
0	0	12	SCRIPT/VS Document Name
12	C	3	Paragraph number of first line
15	F	1	Continued Block Count
16	10	2	Operator Number
18	12	2	CTF Record Length
20	14	5	Read Password
25	19	5	Delete Password
30	1E	45	(Reserved)
75	4B	69	First line of text
144	90	864	Twelve more lines:
144	90	3	Paragraph number of second line
147	93	69	Second line of text
216	D8	3	Paragraph number of third line
219	DB	69	Third line of text
...
936	3A8	3	Paragraph number of thirteenth line
939	3AB	69	Thirteenth line of text
Note: CTF record blocks are not cleared or blanked out. The record length must be read to determine the end of the valid data.			

Using SCRIPT/VS as a Postprocessor

You can use SCRIPT/VS to format reports using data from data processing files. An application program could access these files, perform the necessary computations, and create an output file. The output file could contain GML markup just as if it had been created with normal text-entry procedures. You will then be able to process it with the same flexibility as any of your other documents.

Alternatively, the application program can call SCRIPT/VS as a subroutine. This can be done when the Document Library Facility is installed with SCRIPT/VS. For details on using SCRIPT/VS with the Document Library Facility, refer to the *Document Library Facility Guide*.

You can use SCRIPT/VS to prepare input for itself, except in an ATMS-III environment. For example, you can use the .WF [Write To File] control word to create input files dynamically. These files can later be resubmitted to SCRIPT/VS for further processing.

You can also write formatted output to a file and use it as input for a subsequent invocation of SCRIPT/VS.

Using SCRIPT/VS as a Preprocessor

You can use SCRIPT/VS as a preprocessor if you want SCRIPT/VS to produce an output file that can be processed by some other text formatter or application program. To use SCRIPT/VS as a preprocessor, you must thoroughly understand the text formatter that is to receive the output file prepared by SCRIPT/VS.

Your SCRIPT/VS input file can contain any markup appropriate for SCRIPT/VS (that is, GML tags, control words, macros, and symbols) as well as text and implicit formatting conventions (such as leading blanks, leading tabs, null lines, and full stops). You must build a profile and APFs that interpret the SCRIPT/VS markup and generate appropriate formatter controls.⁰⁰ See Chapter 7, “Processing GML” on page 71 for details about profiles, APFs, and mapping tags to APFs. See Chapter 4, “Processing Symbols” on page 23 for details about symbols. See Chapter 6, “Processing Macros” on page 53 for details about macros.

In most cases, you will find it preferable to use GML markup when using SCRIPT/VS as a preprocessor. The following description, therefore, will assume that your document’s markup observes conventions, such as those described in the *DCF: GML Starter Set Reference*.

Developing Preprocessor APFs and Profiles

SCRIPT/VS has a great variety of general document-handling functions that can be used independently of formatting. You can use these functions to create APFs that will translate a SCRIPT/VS document into suitable input for another program, such as a formatter that can support photocomposers.

For example, the GML starter set APFs for ordered lists and list items automatically generate numbers (or letters) for the items in an ordered list. This is convenient, because it permits the list to be revised without renumbering all the items.

You can create a modified version of the APFs that retains the general processing functions but eliminates the SCRIPT/VS control words that result in formatting. For example, instead of executing the .SK [Skip] and .IN [Indent] control words, you would insert the appropriate formatting controls of the postprocessor into the output stream. The SCRIPT/VS symbol-substitution capability can still be used to calculate parameters for the postprocessor’s formatting controls.

Some of the logical sequence of formatting controls might have to be changed, however. The graphic effect of having the first line of a list item printed to the left of the indentation for the rest of the list item is achieved, in SCRIPT/VS, with the .IN [Indent] control word. The receiving processor might require a different sequence of formatting controls to achieve the same graphic effect.

When modifying an APF in this way, you can structure its logic and function to produce formatting different from that produced by the original APF. You can change the symbol definition for symbols used to achieve different formatting values.

In addition to creating APFs, you would also create a profile that would map to the new APFs. The profile would also issue control words that would turn off justification, page numbering, and so on, so that the output would look like a source file.

```
.pm 0  
.wz off  
.fo off
```

You might also need to translate special characters that might be unacceptable to the postprocessor.

By having two sets of APFs and two profiles, you could continue to print draft copies of the document on a line printer, but could print final output on a photocomposer with the postprocessor.

Redefining Symbols

Many symbols used in source-document markup will not require redefinition. Among these are symbols used in the following ways:

- As abbreviations for lengthy character strings
- As references to generated information that is not format dependent (such as a figure number or section number—but not a page number)
- To enter unkeyable characters that are represented by the same codes in both SCRIPT/VS and the postprocessor.

Handling Directly Entered Control Words

Observing a GML convention for direct entry of control words, such as that described in the *DCF: GML Starter Set Reference*, makes it easy to prepare your document for another processor. The following section describes the specific conventions recommended in that book, but the information is applicable to conventions that can be adopted by your own installation.

Managing a Source Document

The .CM [Comment], .IM [Imbed], and .SE [Set Symbol] control words are executed by SCRIPT/VS before the document is available to the postprocessor. You need take no special action with respect to them.

However, the .RC [Revision Code] and .OC [Output Comment] control words are different; they have a formatting effect. (The .RC control word inserts a revision code character to the left of an output line; the .OC control word places unformatted output comments and carriage control characters into the output.) If the postprocessor has comparable functions, you can define *macros*, called .RC and .OC, to generate the corresponding postprocessor controls. (This technique can be used for all control words if a one-to-one conversion approach is taken.)

If the revision code or output comment functions are not available, you can deactivate them by specifying:

```
.dm rc /.cm
```

or

```
.dm oc /.cm
```

which defines the .RC and .OC macros to be comments.

Preparing for Processing

When you are ready to have SCRIPT/VS prepare your input file for the receiving text processor, take the usual steps needed for SCRIPT/VS execution, as discussed in the SCRIPT command options section of the *DCF: SCRIPT/VS Language Reference*.

Although the file produced by SCRIPT/VS will contain the correct text and markup for your postprocessor, it will not necessarily have the correct physical characteristics. Some postprocessors can require record lengths, formats, or other characteristics that differ from those produced by SCRIPT/VS. You might have to use a utility program, or code your own, to handle such interface requirements.

Appendix B. Compatibility with Earlier Releases of SCRIPT/VS

Appendix B. Compatibility with Earlier Releases of SCRIPT/VS	215
Release 4.0 of the Document Composition Facility	215
The .SY Control Word	217
The .ZZ Control Word	217
Release 3.2 of the Document Composition Facility	217
Release 3.1 of the Document Composition Facility	217
Release 3.0 of the Document Composition Facility	218
Release 3.0 Compatibility with Release 2 of SCRIPT/VS	219
Programming Systems	219
Changes to the SCRIPT Command	219
Changes to SCRIPT Control Words	220
Changed SCRIPT/VS Control-Word Processing	220
Running Headings and Footings in SCRIPT/VS Release 3	223
Historical Background	223
Release 2 Processing	223
Release 3 Processing	224
Release 3 Compatibility	225
Release 3 Enhancements	225
Release 3 Incompatibility	225
Profile Changes	227
SCRIPT/VS Macro Libraries	229
Tabs	229
The SCRIPT/370 Dictionary	229
Unsupported Control Words	229
Changes to SCRIPT Command Options	231
Changes to SCRIPT/VS Control Words	234
Compatibility with TSO/FORMAT	242
Creating a TSO/FORMAT-Compatible Environment	243

Appendix B. Compatibility with Earlier Releases of SCRIPT/VS

This chapter describes the differences between Release 4.0 and earlier versions of SCRIPT/VS:

- SCRIPT/VS Release 3 (Program Number 5748-XX9)
- SCRIPT/VS Release 2 (Program Number 5748-XX9)
- SCRIPT/VS Release 1 (Program Number 5748-XX9)
- SCRIPT/370 Version 3 (Program Number 5796-PHL)
- SCRIPT/370 Version 1 (Program Number 5796-PAF)

Release 4.0 of the Document Composition Facility

- Existing documents created for processing with Release 3.2, 3.1, or 3.0 of SCRIPT/VS will produce comparable results.
- Changes made to DCF for corrective service may cause changes in the formatted output.
- Any user-installation modifications to certain SCRIPT/VS modules will have to be reapplied.
- The default text library has changed from SVTEXT to DSMHYLIB. Release 4.0 uses DSMHYLIB only, it cannot use SVTEXT. DSMHYLIB is used with Release 4.0 only. SVTEXT is used with Release 3.2 only. However, if you have both DSMHYLIB and SVTEXT available at the same time, you can use both Release 4.0 and 3.2.
- Some sort sequences have been changed to adhere to National Language Support. New sort sequences have been added for the new languages. See Chapter 13, "Index Sort Sequence" on page 177 for information about these sort sequences.
- Dictionaries named DAN, FIN, ICE, NOR, POR, and SWE are provided to support the new languages. Any dictionaries that already exist with those names must be renamed.
- There are now algorithmic hyphenators for all the languages SCRIPT/VS supports.
- If the new English algorithmic hyphenators are installed, the algorithmic hyphenation for English American, English Canadian, and English United Kingdom will be different than before. Algorithmic hyphenation is now available for all the languages SCRIPT/VS supports; therefore, the languages supported before this release that did not have algorithmic hyphenators may hyphenate differently. That algorithmic hyphenator will be used if a valid hyphenation point is not found in the dictionary. (Previously, if a valid hyphenation point was not found in the dictionary search, the search was stopped.)

Many items that are shipped with DCF will have new names for SCRIPT/VS Release 4. Table 11 on page 216 describes these items, along with their SCRIPT/VS Release 3 and SCRIPT/VS Release 4 names.

Table 11. Items with Name Changes for DCF Release 4.0		
Description	Release 3.2 Name	Release 4.0 Name
Starter Set Profile	DSMPROF3	DSMPROF4
Bar Code Profile	DSMBPROF	No change
SMFF Profile	DSMFPROF	No change
Schedule Profile	DSMSPROF	No change
Transparency Profile	DSMTPROF	No change
Memo Profile	DSMMPROF	No change
Macro Library	DSMGML3	DSMGML4
Value of macro library symbol	DSMMAC32	DSMMAC40
CMS Dictionaries	SVTEXT	DSMHYLIB
Note: For more information about the following items, see the appropriate Program Directory.		
CMS Shared Segment Module	DSMSEG3	DSMSEG4
CMS Shared Segment Module		DSMSEG4X
CMS Shared Segment Module		DSMSEG4B
CMS Disk Resident Module	DSM3	DSM4
TSO Module	DSMTSS30	DSMTSS40
ATMS Module	DSMATS30	No change
DLF/VSE Module	DSMLDS30	No change
DLF/MVS Module	DSMLOS30	No change
FLIP Module in CMS	DSMCMF30	DSMCMF40
FLIP Module in MVS	DSMBOF30	DSMBOF40
FLIP Module in VSE	DSMLDF30	DSMLDF40
Service EXEC (called by DSMSRV40)	DSMSRVXA	DSMSRVX4
Service EXEC	DSMSRV32	DSMSRV40
CMS Disk Resident Generator	DSMGND32	DSMGND40
CMS Shared Segment Generator	DSMGNS32	DSMGNS40
Dictionary Selector and Module Loader	DSMDCT32	DSMDCT40
Manual Installation EXEC	DSMINS32	DSMINS40
SMFF Shared Segment Generator	DSMGSF32	DSMGNS40
CMS EXEC for creating and modifying user dictionaries	DSMDMP3	DSMDMP4
Dictionary TEXTLIB Generator	DSMTXT32	DSMTXT40

Note: Since the Office Document Feature is not changing for SCRIPT/VS Release 4, the names of its profile (DSMOPROF) and installation exec (DSMINSOD) will remain the same as for Release 3.2.

Table 13 on page 231 shows the changes to the SCRIPT command options, and Table 14 on page 234 shows the changes to SCRIPT control words.

The listed changes are cumulative: Release 4.0 incorporates and includes changes introduced in SCRIPT/VS Release 3, SCRIPT/VS Release 2, SCRIPT/VS Release 1, SCRIPT/370 Version 3, and earlier versions of SCRIPT/370. For details on the functions of individual SCRIPT/VS control words and SCRIPT command options, see the appropriate chapters in this book.

The .SY Control Word

The .SY control word is valid for diagnosis purposes only. Refer to the *Document Composition Facility: Diagnosis Guide and Reference* for information about the .SY control word.

The .ZZ Control Word

The .ZZ control word is valid for diagnosis purposes only. Refer to the *Document Composition Facility: Diagnosis Guide and Reference* for information about the .ZZ control word.

Release 3.2 of the Document Composition Facility

- Existing documents created for processing with Release 3.1 or 3.0 of SCRIPT/VS will produce comparable results.
- Any user-installation modifications to certain SCRIPT/VS modules will have to be reapplied.
- DCF 3.2 supports version 1 of the Country Extended Code Page (CECP) instead of version 0. Support of version 1 of the CECP changes the mapping of a few special characters. See Chapter 13, "Index Sort Sequence" on page 177 for information about the new index sort sequence.

Table 13 on page 231 shows the changes to the SCRIPT command options, and Table 14 on page 234 shows the changes to SCRIPT control words.

The listed changes are cumulative: Release 3.2 incorporates and includes changes introduced in SCRIPT/VS Release 3, SCRIPT/VS Release 2, SCRIPT/VS Release 1, SCRIPT/370 Version 3, and earlier versions of SCRIPT/370. For details on the functions of individual SCRIPT/VS control words and SCRIPT command options, see the appropriate chapters in this book.

Release 3.1 of the Document Composition Facility

- Existing documents created for processing with Release 3 of SCRIPT/VS will produce comparable results.
- Any user-installation modifications to certain SCRIPT/VS modules will have to be reapplied.
- Changes to the logical device table macro have resulted in a reorganization of logical device table format. This requires users who have defined local logical devices to recode and recompile their logical device definitions.
- The sort sequence for special characters and accented characters in index entries has been changed. See Chapter 13, "Index Sort Sequence" on page 177 for information about the new index sort sequence.

Table 13 on page 231 shows the changes to the SCRIPT command options, and Table 14 on page 234 shows the changes to SCRIPT control words.

The listed changes are cumulative: Release 3.1 incorporates and includes changes introduced in SCRIPT/VS Release 3, SCRIPT/VS Release 2, SCRIPT/VS Release 1, SCRIPT/370 Version 3, and earlier versions of SCRIPT/370. For details on the functions of individual SCRIPT/VS control words and SCRIPT command options, see the appropriate chapters in this book.

Release 3.0 of the Document Composition Facility

- Existing documents created for processing with Release 2 of SCRIPT/VS will, with few exceptions, produce comparable results when formatted with Release 3.
- Documents created for processing with Release 1 of SCRIPT/VS might require markup changes when formatted with Release 3.
- The OS/VS1 environment is not supported in Release 3 of the Document Composition Facility.
- The algorithms for horizontal justification have been improved. This might cause differences in the placement of text on a line.
- The algorithms for vertical placement of text on the page have been improved. This might cause differences in the placement of lines on a page.
- Several restrictions regarding the use of widow zones with inline keeps and footnotes have been removed. This might cause differences in the placement of lines in a column.
- Several errors and inconsistencies in box handling found in Release 2 have been corrected for Release 3. The major area corrected deals with the drawing of horizontal rules for boxes.
- Changes to the logical device table (LDT) macro have resulted in a reorganization of the logical device table format. This requires users who have defined local logical devices to recode and recompile their logical device definitions.
- Trailing colons in SCRIPT/VS macro definitions should no longer be used when storing the definitions in a macro library. The extra colon will be treated as text.
- The DUMP option of the SCRIPT/VS command and the .ZZ control word are no longer provided.
- The .RD [Read Terminal] control word no longer performs a break or a section break.
- The .RH [Running Heading], .RF [Running Footing], and .FN [Footnote] control words no longer preexecute any control words or tags when the running headings and footings and footnote leader are defined. Definitions are saved in their entirety and executed completely on each subsequent output page. Running headings and footings can include most control words, macros, and GML tags.
- The .HY [Hyphenate] control word has new parameters whose defaults might disqualify hyphenation points that were used in Release 2. The same frequency of hyphenation can, however, be achieved in Release 3 with correct parameter settings.

- Reference numbers, as well as the UNFORMAT and NUMBER options of the SCRIPT command, are available only on line printers.
- The initial values have changed for the control words relating to margins.
- Deferred control word placement and page number referencing (the .PT, .PI, and .SE control words, which reference the page number) have been improved. The change requires that there be text after these control words before they can be processed.
- Delay imbeds (.DI) can no longer be executed in running headings and footings.

Table 13 on page 231 shows the changes to the SCRIPT command options, and Table 14 on page 234 shows the changes to SCRIPT control words. Unsupported control words and the control words that provide a comparable function are listed in Table 12 on page 230.

The listed changes are cumulative: Release 3 incorporates and includes changes introduced in SCRIPT/VS Release 2, SCRIPT/VS Release 1, SCRIPT/370 Version 3, and earlier versions of SCRIPT/370. For details on the functions of individual SCRIPT/VS control words and SCRIPT command options, see the appropriate chapters in this book.

Release 3.0 Compatibility with Release 2 of SCRIPT/VS

With the exceptions noted below, existing documents created for processing with Release 2 of SCRIPT/VS will produce comparable results when formatted with Release 3.

Documents created for processing with Release 1 of SCRIPT/VS might require markup changes when formatted with Release 3.

Programming Systems

Release 3 of the DCF will not be provided for the OS/VS1 environment.

Changes to the SCRIPT Command

For Release 3, the following changes have been made to the SCRIPT command:

- The CHARS and DEVICE options have been changed to reflect the addition of the 4250 printer, the 3800 Printing Subsystem Model 3, and the 3820 Page Printer.
- The UNFORMAT and NUMBER options have been changed so that they are ignored for page printers.
- The PRINT option has been changed to take advantage of virtual 3800 support in the VM/370 System Product.
- The FONTLIB option has been added to identify font libraries for use with page printers.
- The SEGLIB and NOSEGLIB options have been added to identify segment libraries for use with page printers.
- The NOCONT option has been added to stop formatting when an error is encountered.

- The DUMP option is now invalid and will no longer be supported.
- The SEARCH option is now valid in CMS and can be used to specify file type for imbed, append, and .DD files in CMS.

Table 13 on page 231 shows a complete list of the changes to SCRIPT command options.

Changes to SCRIPT Control Words

Table 14 on page 234 shows the changes made to individual control words. Other differences are discussed below.

Changed SCRIPT/VS Control-Word Processing

The Release 2 functions of several existing control words have changed. The changes include:

- The default style of page numbering has been changed. In Release 2, all pages except the first were numbered with the following control word sequence:

```
.rt top all 1 ///page &
```

This caused 'PAGE n', where 'n' is the page number, to appear in the upper right-hand corner of all pages except the first one. In Release 3, the method of numbering pages has been changed to:

```
.rf on
.sp 2
.ce - & -
.rf off
```

This change will cause '- n -', where 'n' is the page number, to appear at the bottom center of each page. By using the procedure described in the example given in the following bulleted item, you can restore the Release 2 method of page numbering.

If page numbering is turned off (.PN OFF), the string '- n -' will be centered at the bottom of all pages because of the default running footing. This can be avoided by redefining the running footing or by canceling it. With Release 2, you will get 'PAGE' in the upper right-hand corner of the page if the page numbering is turned off.

- The initial values for the following control words have been changed:

```
.BM [Bottom Margin]
.FM [Footing Margin]
.FS [Footing Space]
.HM [Heading Margin]
.HS [Heading Space]
.PM [Page Margin]
.TM [Top Margin]
```

The initial values for .HS, .HM, .FS, and .FM are zero. The initial values for .TM, .BM, and .PM are given in "Logical Device Table" in the *DCF: SCRIPT/VS User's Guide*.

However, the Release 2 initial values can be restored by including the following control words in the document profile:

```
.tm +3
.hm 2
.hs 1
.bm +3
.fm 2
.fs 1
.rh cancel
.rf cancel
```

When restoring the Release 2 values, it is necessary to cancel the default running heading and running footing.

It is recommended that existing documents that use running titles be converted to use the more flexible running heading and footing facilities, which provide full text-composition function.

- The default binding has been changed from the font-dependent spacing of two characters, to a fixed width of one inch. You can restore the former spacing if necessary by including the following control word line in the profile:

```
.pm 2 nobind
```

It is recommended, however, that you use the new, fixed one-inch binding or some other fixed-width binding rather than character spacing.

- The syntax of the .LS [Line Spacing] control word has been changed. The SCRIPT/370 IUP syntax will be recognized. But it is recommended that existing documents that use the unsupported form of .LS be converted to use the Release 3 syntax. For example, a document containing:

```
.ls 1
```

can be modified to contain:

```
.ls normal 2
```

- The .RD [Read Terminal] control word will no longer cause a break or a section break. It is recommended that existing documents that use the non-STOP functions of .RD be converted to use the .RV [Read Variable] or .TE [Terminal Input] control words. However, the non-STOP function can be simulated with the .SC [Single Column Mode] and .MC [Multicolumn Mode] control words. For example, a document containing:

```
.rd 1
```

can be modified to contain:

```
.sc
.rd
.mc
```

- The .HY [Hyphenate] control-word parameters MINWORD and MAXPT are new for Release 3, and they have no equivalent in Release 2. The default values for these parameters might disqualify hyphenation points that were used in Release 2. You can achieve the same frequency of hyphenation in Release 3 by setting MAXPT to 2, MINPT to 2 and MINWORD to 4.
- If you use GML tags to start and end a running heading or running footing definition, this definition might not be processed correctly in Release 3 because of to changes in running heading and running footing processing. To avoid this problem you must do one of the following:

- Change the GML tags (your markup) to SCRIPT/VS control words that provide the same formatting
- Create alternative APFs for your GML tags.

Refer to “Running Headings and Footings in SCRIPT/VS Release 3” for more details.

Several restrictions have been lifted and improvements made that can cause differences in the placement of text on a page. For example,

- Inline keeps no longer automatically end a previous keep.
- Footnote placement improvements have been made, especially with regard to the handling of deferred control words such as .PI [Put Index] and .PT [Put Table of Contents].
- The horizontal justification algorithms have been improved.
- The text distribution algorithms have been improved.
- In Release 2.0, word space on the 3800 Printing Subsystem was always a 15-pitch space, even though the body was 10- or 12-pitch; this resulted in a very tightly set line. In Release 3, the normal word space is the same as the body font.

Furthermore, the GML Starter Set uses .HY RANGE to implement the new hyphenation and justification algorithm to reduce the number of words that are hyphenated. This might result in a greater number of output pages than produced using Release 2.0. If this is unacceptable for your purposes, you should use the following control words:

- .ws normal p5: where p5 is approximately 1/15 of an inch
- .hy range 1.0: to indicate that word-space sizes are not to be varied.

Note: The values specified for the RANGE parameter of the .HY [Hyphenate] control word in the GML starter set might result in a very tightly set line if certain fonts are used. If this is unacceptable for your purposes, modify your own copy of profile DSMPROF3 to specify larger compression and expansion values, for example, '.hy range 1.0 1.5'.

- In Release 2 GML processing, sometimes leading blanks on input lines were retained in the input and sometimes they were discarded. In Release 3, leading blanks will always be retained in the input.
- Boxes (.BX) are built differently in DCF Release 3.0 from the way they were in Release 2.0. This difference is noticeable when there is an attempt to change the box characters used in creating boxes with the .TR [Translate Character] control word. In Release 2, you could change the characters used in constructing boxes by translating the box characters with the .TR control word. In Release 3, output translation is done *before* boxes are constructed. Therefore, .TR can no longer be used to translate box characters. Appendix B describes how to define additional box character sets. The .BX control word CHAR parameter allows specification of a particular box character set. Refer to the description of the .BX [Box] control word in the *DCF: SCRIPT/VS Language Reference* for more information.

Running Headings and Footings in SCRIPT/VS Release 3

The most significant external change to SCRIPT/VS Release 2 functions made in Release 3 is in the definition and processing of running headings and footings. These notes describe the nature and extent of the changes and provide suggestions to current users for implementing them.

Historical Background

In Release 2 of DCF running headings and running footings were processed in two distinct and separate phases:

- Definition phase: When a running heading or footing is defined, some control words in the definition are executed immediately, using the current environment. Symbol substitution is normally performed during the definition phase. Any text in the definition, and all control words that are not executed immediately, are saved for later processing.
- Execution phase: When each subsequent page is started, the text and control words saved during the definition phase are processed in a special mode. This special mode does not include symbol substitution, GML processing, or macro execution.

Release 2 Processing

The simple running headings and footings that typical users created could contain only text and a few basic control words. For example:

```
.rh on
.bf GB12
.ce Internal Use Only
.pf
.sp
.rh off
```

The class of control words that were saved for execution phase were defined to include all of the formatting-related control words that such users were likely to include.

However, text programmers used the facilities of SCRIPT/VS that were not formatting related to implement GML tags and to provide other complex formatting for their users. In doing so, the following problems became apparent:

- Running headings and footings often must include symbols. In Release 2 this could be accomplished only with such complex formatting as:

```
.su off
.rf on
.sp
.ec .su .sx f /&title.//Page &/
.rf off
.su on
```
- Running headings and footings that perform logic as well as formatting must include control words that are not formatting related, such as .IF, .THEN, and .OR. In Release 2 this can be accomplished only with such complex formatting as:

```

.su off
.rf odd
.sp 2
.ec .if &E eq &0
.ec .th .su .ri &head.: &0
.ec .el .su .ri &head.: &E., &0
.rf off
.su on

```

- Running headings and footings that perform logic as well as formatting must often include iterative processing. In Release 2 this was not possible.
- Running headings and footings must often invoke macros. In Release 2 this was not possible.
- In documents marked up with GML, running headings and footings must often contain GML tags. It was not possible to include GML in Release 2.
- Release 2 running heading and footing processing is extremely difficult. Knowledge of the internal design of SCRIPT/VS is required to use running headings and footings effectively.

Release 3 Processing

Running headings and footings have been reimplemented for Release 3. Primary design criteria for the new implementation were:

- Remove all Release 2 restrictions
- Maintain compatibility where possible.

In Release 3, running headings are still processed in two distinct phases, but the processing performed during each phase has changed:

- Definition phase: The entire heading or footing, including all text, symbols, macros, and GML tags, is saved for later processing.⁸ An .RH, .RF, .FN, or .HN⁹ control word will end the definition phase. Also, if you invoke a macro whose name begins with the characters "rh", "rf", "fn", or "hn" and have not also specified the .EM [Execute Macro] control word, the definition phase will be ended. No control words are executed during definition phase. Symbol substitution and GML processing are not performed during definition phase.
- Execution phase: The saved definition is executed in ordinary macro mode. Only control words that result in a page eject are disallowed.

This implementation meets both of the primary design criteria:

- All of the functional deficiencies listed above are alleviated. Running headings and footings can contain text, symbols, macros, logical functions, iterative processing, and GML markup.
- Compatibility for existing definitions has been maintained in almost all cases.

⁸ To ensure that an entire document does not become a heading or footing if the user forgets to enter ".RH OFF" or "RF OFF", definitions are terminated at the end of the current file or macro.

⁹ The .HN control word is an unsupported control word. See Table 12 on page 230 for more information.

Release 3 Compatibility

Release 3 is completely compatible with Release 2 for most running heading and footing definitions. All of the examples cited above, which are typical of the majority of actual definitions, will be processed correctly in Release 3 *without change*.

The functional enhancements to running headings and footings in Release 3 will allow sophisticated definitions, such as the last two examples above, to be implemented much more simply. The second example above can now be defined simply as:

```
.rf on
.sp
.sx f /&title.//Page &/
.rf off
```

The third example can now be defined simply as:

```
.rf odd
.sp 2
.if &E eq &0 .ri &head.: &0
.el .ri &head.: &E., &0
.rf off
```

Note: The examples given for Release 2 will work with Release 3 without being changed. However, if these examples are coded with .SU OFF/.SU ON inside the running footing definition, then errors will result because symbol substitution is turned off (by the .SU OFF) during the execution phase of the running footing.

Release 3 Enhancements

The functional enhancements in Release 3 will enable GML users and text programmers to create running heading and footing definitions that were not possible in Release 2.

For example, many technical and professional style guides require that all footnotes for each chapter be formatted together at the end of the chapter or article. Such a document might require that the running footing for the footnote portion of the document contain a reference to the chapter heading:

```
.rf on
.sp
.ri (Notes for :hhref refid=chap1.)
.rf off
```

Release 3 Incompatibility

One form of running heading definition that might have been used in Release 2 will require some modification to work in Release 3. If the running heading (or a running footing) definition is performed as the subject of one of the logic control words (.IF, .TH or .EL) that definition will not be accepted in Release 3. For example, the control word sequence:

```
.if 1 lt 2
.th .rh on
.th .ce Page &
.th .rh off
```

creates the desired heading in Release 2 but causes an error message in Release 3. To work correctly, the control word sequence must be changed to bypass the running heading definition:

```
.if 1 lt 2
.el .go next
.rh on
.ce Page &
.rh off
...next
```

Another form of running heading definition that some current users might use will not be processed correctly in Release 3. This technique uses GML tags to start and to end the definition, rather than as a part of the definition. For example:

```
.aa rh rhon rhoff
.dm rhon /.rh on
.dm rhoff /.rh off
.dc gml : :e
.gs tag on
.ms on
:rh
.bf GB12
.ce Internal Use Only
.pf
.sp
:erh
```

Users who have used this technique in defining their running headings and footings must change either their documents or their profiles before using Release 3. Otherwise, their running heading definitions will be incorrect, and, for example, the text and controls between the sample :RH and :ERH tags can be formatted in the body of the document.

The simplest way of coping with this incompatibility is to change the markup in the affected documents. The sample :RH and :ERH tags can be replaced with the .RH control word:

```
.rh on
.bf GB12
.ce Internal Use Only
.pf
.sp
.rh off
```

The main advantage of this approach, in addition to simplicity, is that the definition can later be enhanced to take advantage of new Release 3 functions.

If imbed files are used in running headings and footings and those files contain characters that need to be translated (for example, by the .TI [Translate Input] control word), then the translate control words must also be entered within the running heading and footing. This is so because control words such as .IM [Imbed] are not processed until the execution phase, when the running heading and footing are formatted using the initial formatting environment.

Profile Changes

When changing the markup in documents with this incompatibility is either impractical or undesirable, alternative APFs for the sample :RH and :ERH tags would have to be provided.

First, the APF-to-tag association must be provided. For purposes of illustration, we will assume that the tags are named :RH and :ERH and that the macros are named RHON and RHOFF. Therefore:

```
.aa rh rhon rhoff
```

must be added to the profile.

In the following set of APFs, the lines of the running heading definition will be captured with the Release 3 .PX control word and saved in a macro. When the entire definition has been captured, a running heading will be defined that invokes the macro.

In two cases the macros described below will not work correctly: when the :RH tag is followed immediately with the :ERH tag or with a control word. In each case, inserting a comment line (.) after the line with the :RH tag on it will cause the macros to process correctly.

An alternative RHON macro might be defined as follows:

```
.dm rhon on
.se flag = off
.dm head off
.gs scan line
.if &L'&line gt 0 .dm head() /&V'&line
.su off
.gs tag off
.px null /.save
.dm off
```

This code performs the following processing:

1. Starts the definition of the RHON APF.
2. Initializes a flag to 'off' to indicate the beginning of processing in the macro.
3. Creates a new macro called HEAD to hold the lines of the running heading and to ensure that one is not already defined.
4. Gets any residual text following the :RHON tag into a symbol.
5. Adds the line to the HEAD macro, if there is one.
6. Delays symbol substitution until the definition is processed.
7. Turns off GML tag processing.
8. Prefixes all subsequent input lines so that they are passed to the SAVE macro for processing.
9. Ends the definition of the RHON APF.

The SAVE macro might be defined as follows:

```

.dm save on
.if /&flag eq /on .me &*
.se *i = index &U'&*. :ERH
.if &*i ne 0 .se *rest = substr &* &*i
.th .se *i = &*i-1
.th .se * = substr &* 1 &*i
.su .dm head() /&V'&*
.if &L'&*rest eq 0 .me
.su on
.gs tag on
&*rest
.dm off

```

This code performs the following processing:

1. Defines the SAVE macro, which places input lines into the HEAD macro.
2. Tests the &flag symbol. If processing is near completion, ends the macro immediately and puts back the text passed to it.
3. Scans the line to see if it contains the end tag :ERH.

Note: If the tag is named something other than :RH and :ERH, you will need to adjust this line to contain the correct end tag name.

4. If the line contains the end tag being looked for, it splits the line. The end tag and the text following it are put into the &*rest symbol.
5. If the line contains the end tag, the code calculates the position of the last character before the end tag by subtracting 1 from &*i.
6. If the line contains the end tag, the code sets the &* symbol to contain only the part of the line before the end tag.
7. Puts the text onto the next line of the HEAD macro, with substitution on only for this line.
8. If the symbol &*rest has not been set, it ends the macro.
9. Turns on symbol substitution.
10. Turns on GML tag processing.
11. Processes the &*rest symbol, which contains the end tag.
12. Ends the definition of the SAVE macro.

A companion RHOFF macro might be defined as follows:

```

.dm rhoff on
.se flag = on
.px off
.rh on
.head
.rh off
.dm off

```

This code performs the following processing:

1. Defines the RHOFF macro to handle the :ERH tag.
2. Turns the flag on to indicate processing complete.
3. Turns input line prefixing off.¹⁰
4. Defines the running heading.
5. Uses the HEAD macro created in the SAVE macro.

¹⁰ The flag is necessary because the next line after the :ERH might already have been prefixed as part of the search for the residual text of the :ERH tag. If this happens, the SAVE macro will be executed once too often.

6. Ends the running heading definition.
7. Ends the RHOFF APF.

SCRIPT/VS Macro Libraries

Doubling trailing colons in SCRIPT/VS macro definitions will no longer be necessary when storing them in a macro library. The second colon will be printed as text.

Tabs

In Release 3 the maximum number of tabs that can be set at any one time is 64, rather than 99 as it was in Release 2.

The SCRIPT/370 Dictionary

SCRIPT/370 supported a hyphenation-exception dictionary called SCRIPT XDICT. The XDICT dictionary was used to determine how to hyphenate words that were not correctly hyphenated by the hyphenation algorithm. The user could create and modify his own hyphenation exception dictionaries using the HYPEDIT command.

SCRIPT/VS does not support either the HYPEDIT command or the exception dictionaries. Instead, SCRIPT/VS provides dictionaries that support both spelling verification and automatic hyphenation in nine languages.

You can also create and update a temporary dictionary for use when your document is being formatted, called the addenda dictionary, by using the .DU [Dictionary Update] control word.

Unsupported Control Words

The following table lists unsupported SCRIPT/VS control words. SCRIPT/VS will continue to recognize these control words, but it is recommended that you use the new control words for these functions.

Note: The .SY and .ZZ control words are recognized but not supported and are reserved for diagnostic purposes.

Table 12. Unsupported Control Words	
Unsupported Control Word	Control Word Providing Comparable Function
.BT	.RF [Running Footing]
.CO	.FO [Format Mode]
.CW	.DC [Define Character]
.DI	.DA [Define Area]
.EB	.RF [Running Footing]
.EP	.PA [Page Eject]
.ET	.RH [Running Heading]
.EZ	.DH [Define Head Level]
.FI	.FO [Format Mode]
.FM	.RF [Running Footing]
.FS	.RF [Running Footing]
.FT	.RF [Running Footing]
.HE	.RH [Running Heading]
.HM	.RH [Running Heading]
.HN	.RH [Running Heading]
.HS	.RH [Running Heading]
.NB	.BC [Balance Columns]
.NC	.FO [Format Mode]
.NJ	.FO [Format Mode]
.OB	.RF [Running Footing]
.OP	.PA [Page Eject]
.OT	.RH [Running Heading]
.PP	.SK [Skip]
.PS	.DC [Define Character]
.RT	.RH [Running Heading]
.SL	.LS [Line Spacing]
.TT	.RH [Running Heading]

Changes to SCRIPT Command Options

The following table is a summary of changes to the SCRIPT command options. The codes used in this table are:

New	The command option was introduced in the indicated product.
Changed	The command option was changed in the indicated product
V/3	The command option was introduced or changed in SCRIPT/370 Version 3.
R.1	The command option was introduced or changed in SCRIPT/VS Release 1.
R.2	The command option was introduced or changed in SCRIPT/VS Release 2.
R.3	The control word or option was introduced or changed in Release 3.
R.3.1	The command option was introduced or changed in Release 3.1.
R.3.2	The command option was introduced or changed in Release 3.2.
R.4	The command option was introduced or changed in Release 4.0.
Invalid	The command option is no longer valid. Its function is performed by a new control word command option in SCRIPT/VS. SCRIPT/VS does not accept or process the old control word or option.

Table 13 (Page 1 of 3). Changes to SCRIPT Command Options		
Option	Code	Changes
2PASS	Invalid	Specify TWOPASS instead.
ADJUST	Invalid	Specify BIND instead.
ADJUSTnn	Invalid	Specify BIND instead.
BIND	New R.1	Shift the page to the right.
CENTER	Invalid	Specify BIND instead.
CENTERnn	Invalid	Specify BIND instead.
CHARS	New R.1	Specify up to four fonts (Valid only for the 3800 Printing Subsystem).
	Changed R.3	Specify more than four external font identifiers for page printer logical devices; however, formatting is <i>not</i> limited to the fonts so specified.
	Changed R.3.2	Specifies the AFM file for PostScript devices.
CTF	New R.2	Prepare output in STAIRS/VS Condensed Text Format. Mutually exclusive with PRINT, FILE, and TERM.
	Changed R.3	If any logical device other than the STAIRS logical device is specified, the CTF option is ignored.
DDUT	New R.3	Allows the redefinition of a SCRIPT/VS utility file to a nonutility file.
DEBUG	Invalid	Specify NOSPIE instead.
DEST	New R.1	Specify a remote output station.

Table 13 (Page 2 of 3). Changes to SCRIPT Command Options

Option	Code	Changes
DEVICE	New R.1	Specify a logical output device.
	Changed R.2	STAIRS logical device added.
	Changed R.3	New logical devices added to describe the 4250 printer, the 3800 Printing Subsystem Model 3, and the 3820 Page Printer.
	Changed R.3.2	New logical devices added to describe PostScript devices and to allow generic specification of page printers.
DUMP	Invalid	No longer supported.
FILE	Changed R.1	Specify the name of the output file.
	Changed R.2	Mutually exclusive with TERM, PRINT, and CTF. Default logical device is 1403W6.
FONTLIB	New R.3	Identify a library containing font information.
FPASSES	New R.3.1	Prepare with multiple formatting passes and produce output on the last pass.
LIB	New R.1	Specify up to eight library names in CMS, one in TSO.
MARK	Invalid	No longer supported.
MESSAGE	New R.1	Control the timing and destination of messages.
NOCONT	New R.3	Terminate formatting if an error is encountered.
NODDUT	New R.3	Disallows the redefinition of a SCRIPT/VS utility file to a non-utility file.
NOPROF	New V/3	Suppress the PROFILE option.
NOSORCDD	New R.4	Explicitly specifies that the primary input file is a data set name.
NOSEGLIB	New R.3	Suppress the segment library.
NUMBER	Changed R.3	Ignored for page devices.
NUMBERnn	Invalid	Specify NUMBER instead.
OFFLINE	Invalid	Specify PRINT instead.
OPTIONS	New R.1	Specify a file that contains additional SCRIPT command options.
PAGE	New R.1	Specify one or more ranges of pages to print, or request SCRIPT/VS to prompt you to enter page number ranges.
PAGEnnn	Invalid	Specify PAGE(nnn) to print from page nnn.
PRINT	Changed R.2	Mutually exclusive with TERM, FILE, and CTF.
	Changed R.3	In CMS, will take advantage of virtual 3800 support in the VM/370 System Product. Not valid for 4250 printer logical devices.
	Changed R.4	In CMS, AFP output files will go directly to SPOOL saving processing time and DASD.
PROFILE	Changed R.1	Specify the name of a profile.
	Changed R.2	Part of the profile can be executed after the primary input file is finished, providing an epifile facility.
PSOUT	New R.3.2	Used to specify ASCII or EBCDIC output files for PostScript devices configured to accept 8-bit ASCII.
SEARCH	New R.1	Specify a library to be searched for imbedded files. (Not valid in CMS.)
	Changed R.3	Specify a filetype for imbed, append, and .DD files in CMS.
SEGLIB	New R.3	Identify a page segment library containing page segment information.

Table 13 (Page 3 of 3). Changes to SCRIPT Command Options		
Option	Code	Changes
SEPMASSTR	New R.4	Specify which masters to create.
SINGLE	Invalid	Specify PAGE (nn ONLY) to print a single page, nn.
SORCDD	New R.4	Notifies SCRIPT to process the primary input file as a ddname.
SPELLCHK	New R.1	Enable the .SV [Spelling Verification] control word to perform spelling verification.
SYOFF	New R.3	Disables the .SY [System Command] control word. SYOFF is the default. Applicable only in the CMS and TSO environments.
SYON	New R.3	Enables the .SY [System Command] control word after SYOFF is specified. Applicable only in the CMS and TSO environments.
STOP	Changed R.2	Sheet alignment has been simplified.
SYSVAR	New V/3	Set Symbol values from the command options.
TERM	New R.1	Display formatted output at the terminal.
	Changed R.2	Mutually exclusive with PRINT, FILE, and CTF.
TLIB	New R.2	Specify libraries containing spelling verification and hyphenation dictionaries (Valid only in CMS).
TRANSLATE	Invalid	Specify UPCASE to have lowercase characters translated to uppercase.
TWOPASS	New R.1	Prepare with two formatting passes, and produce output on the second pass.
UNFORMAT	New V/3	Process the .IM [Imbed], .AP [Append], and .EF [End of File] control words, and read lines from imbedded files to include in the unformatted listing. Symbol substitution is performed, but the input line is printed as entered.
	Changed R.3	Ignored for page devices.
UPCASE	New R.1	Fold all lowercase letters to uppercase letters before printing.

Changes to SCRIPT/VS Control Words

The following table is a summary of changes to the SCRIPT/VS control words. The codes used in this table are:

New	The control word was introduced in the indicated product.
Changed	The control word was changed in the indicated product.
V/3	The control word was introduced or changed in SCRIPT/370 Version 3.
R.1	The control word was introduced or changed in SCRIPT/VS Release 1.
R.2	The control word was introduced or changed in SCRIPT/VS Release 2.
R.3	The control word was introduced or changed in Release 3.
R.3.1	The control word was introduced or changed in Release 3.1.
R.3.2	The control word was introduced or changed in Release 3.2.
R.4	The control word was introduced or changed in Release 4.0.
Invalid	The control word is no longer valid. Its function is performed by a new control word or option in SCRIPT/VS. SCRIPT/VS does not accept or process the old control word or option.

Table 14 (Page 1 of 9). Changes to SCRIPT/VS Control Words

Control Word	Code	Description
...	Changed R.1	Labels are allowed in SCRIPT/VS macros.
.AA	New R.2	Defines the mapping between GML tags and APFs, and it specifies the rules for scanning the tag.
.AM	New R.3.1	Specifies the value to be added to the current page margin value.
.AN	New R.2	Works in conjunction with .IF and .OR to conditionally process text and control words.
	Changed R.3.2	Allows querying for a PostScript device (PS).
.AP	Changed R.1	Up to 14 tokens can be passed to the appended file. The tokens are not reset when a macro is called. &0 contains the number of tokens passed.
	Changed R.2	An external file name can be specified in parentheses.
.AR	New R.3	Causes subsequent text to be formatted into a <i>named</i> area.
.BC	Changed V/3	Operands ON and OFF restore and cancel column balancing.
.BF	New R.1	Specifies the font in which subsequent text is to be formatted.
	Changed R.2	Saves the current font before beginning a new font.
	Changed R.3	More than one font can be given; the first valid font listed will be used.
.BL	New R.3	Is generated by SCRIPT/VS and executed whenever a blank line is processed.
.BM	Changed V/3	The bottom margin can be specified as an increment to or a decrement from the current value.
	Changed R.1	Default based on the logical device. Changes always take effect on the next page.
	Changed R.3	Default values changed.

Table 14 (Page 2 of 9). Changes to SCRIPT/VS Control Words

Control Word	Code	Description
.BX	New V/3	Draws automatic boxes.
	Changed R.1	New options for drawing a box within a box, for drawing fragments of boxes, and for drawing parallel boxes.
	.Changed R.2	New option for drawing untopped boxes. Vertical rules always overlay text.
	Changed R.3	New parameters to specify that vertical rules are to be placed flush against the edges of the column and to use rules defined with .DR [Define Rule].
	Changed R.4	New parameter allows shading of boxes on AFP printers and PostScript devices.
.CC	Changed R.1	Ejects to a new column only when not already at the top of a column.
.CD	Changed V/3	You can define up to nine displacements for columns, even if you initially specify only one column. The remaining displacements are used when you later increase the number of columns.
.CE	Changed V/3	Accepts input text as a parameter.
.CG	New R.3	Specifies the copy group that PSF should use.
	Changed R.4	Allows more than one Include Medium Map (IMM) to be created between pages and to allow an IMM to be created after the last page.
.CL	Changed V/3	The column line length can be specified as an increment to or a decrement from the current value.
	Changed R.2	The column line length can be specified as an increment to or a decrement from the current value.
.CP	Changed R.1	Ejects to a new page only when not already at the top of a page.
.CT	New R.2	Concatenates a line to the previous input line with no intervening word space.
.DA	New R.3	Defines a <i>named</i> area.
	Changed R.4	Allows shading of <i>named</i> areas on AFP printers and PostScript devices.
.DC	New R.1	Specifies characters with special meaning to SCRIPT/VS.
	Changed R.2	New parameters indicate how to treat special characters in the index.
.DD	New R.1	Maps external file names to SCRIPT/VS file IDs.
	Changed R.2	External file names can contain blanks and other special characters.
.DF	New R.2	Defines internal fonts, which can be composed of underscoring, overstriking, capitalization, font changes (only for the 3800), and pauses for typing element changes (only for typewriter output).
	Changed R.3	New parameters allow you to describe fonts for page printers. New parameter allows you to specify a list of fonts.
	Changed R.3.2	No new parameter, but now allows the definition of PostScript fonts. Rotation is ignored for a PostScript device.
	Changed R.4	New parameter (SEPMASR) allows you to specify if the font is to be included in a separation master.
.DH	New V/3	Specifies formatting parameters for seven levels of automatic headings.
	Changed R.1	You can specify a font for each head level.
	Changed R.2	New parameters allow you to specify the font of the table of contents entry, automatic heading numbering, and so on.
	Changed R.3	New parameters indicate how the heading is to be aligned, formatted, and hyphenated.

Table 14 (Page 3 of 9). Changes to SCRIPT/VS Control Words

Control Word	Code	Description
.DL	New R.2	Specifies the language to be used for spelling verification and hyphenation.
	Changed R.3.1	Specifies the language to be used for the index sort sequence.
	Changed R.4.0	Six new languages added.
.DM	New V/3	Allows user-written functions to extend or replace SCRIPT control words.
	Changed R.1	You can specify a macro with more than one input line and can store macros in a library. When a macro is invoked, SCRIPT/VS sets local symbols &*0 through &*n (n being the number of tokens passed to the macro; &*0 contains the number of tokens passed). The macro can set local symbols that begin with &*.
	Changed R.3	Macros can be defined inline without prefixing each line of the macro with a .DM [Define Macro] control word.
.DR	New R.3	<i>Named</i> rules of varying thickness can be defined for page printers.
	Changed R.4	New parameter (SEPMASTR) allows you to specify if the rule is to be included in a separation master.
.DU	New R.1	Temporarily adds words to the spelling verification and hyphenation dictionary.
	Changed R.2	Specifies which dictionaries will be searched, and in what order.
.DV	New R.3	Defines and assigns a value for text variables.
.EC	New R.1	Executes a control word, even if there is a macro of the same name.
.EF	Changed V/3	CLOSE operand allows you to suspend an input file.
	Changed R.2	When .EF CLOSE is used in the profile, the remainder of the profile is processed as an epifile after the primary input file has finished.
.EL	New R.2	Executes only if the most recently executed .IF, .AN, or .OR was false.
.EM	New R.1	Executes a macro, even if macro substitution is off.
.EQ	New R.3	This is a reserved control word.
.ES	New R.3	Controls the formatting of multiple blanks by specifying the width of the extra spaces.
.FL	New R.2	Specifies that blocks of text are to be kept together and placed at the top or bottom of subsequent columns or pages.
.FN	New V/3	Specifies text to be placed at the bottom of the current page.
	Changed R.1	New parameter: LEADER allows you to define leading text lines to precede the first footnote of each page. Column balancing occurs on pages with footnotes. Footnotes are formatted to the line length instead of to the column width.
	Changed R.2	Footnotes can be of arbitrary size and will be divided into pieces of one-quarter page depth at most. Changes to the footnote always take effect on the next page.
.FO	New V/3	Operands ON and OFF allow you to restore and cancel formatting (concatenation and justification).
	Changed R.1	New parameters: LEFT, RIGHT, CENTER, EXTEND, FOLD, and TRUNC.
	Changed R.3	New parameters indicate inside (towards presumed binding) or outside (away from presumed binding) alignment.
.FV	New R.3	Specifies whether text is to be justified. Also can be used to specify if text is to be at the top, bottom, or center of a column if it is not justified.
.GO	New V/3	Branches to a labeled line.
	Changed R.1	Allowed in SCRIPT/VS macros.

Table 14 (Page 4 of 9). Changes to SCRIPT/VS Control Words

Control Word	Code	Description
.GS	New R.2	Provides a variety of facilities of use in writing GML APFs.
	Changed R.3	The COPY parameter allows a GML APF to examine the residual text following a GML tag without disturbing it. The INSERT parameter allows a GML APF to insert text where the tag was. New parameters: ATTVAL retrieves the value of an attribute and puts the value into a symbol. OPTVAL retrieves the value of user-defined options specified in the SCRIPT command and puts them in a symbol.
.HR	New R.3	Places a horizontal rule in the current column.
.HY	New V/3	Enables hyphenation.
	Changed R.1	Hyphenation is performed using a dictionary.
	Changed R.2	An algorithmic hyphenator can be used in place of or in conjunction with multiple dictionaries.
	Changed R.3	The maximum number of successive lines to be hyphenated can be specified. The minimum and maximum number of characters preceding and following a hyphenation point and the minimum size of a word available for hyphenation can be specified. The minimum and maximum range by which word spaces can be compressed or expanded to avoid hyphenation can also be specified.
	Changed R.4	Changes the way the ALT parameter is processed.
.Hn	New V/3	Head-level control words for head levels 0 through 6. The default characteristics of each head level are changed when you enable EasySCRIPT control word, and are restored when you disable EasySCRIPT (with the .EZ OFF control word).
	Changed R.3	New parameters of .DH [Define Head Level] provide the same formatting options for headings that are available for text with .FO [Format Mode].
.IC	New R.3	Controls the insertion of extra space between characters for purposes of horizontal justification.
.IE	New R.2	Used in formatting automatic indexes.
.IF	New V/3	Conditionally processes text and control words.
	Changed R.2	The comparands are no longer limited to 8 characters and will be evaluated even if symbol substitution is off.
	Changed R.3.2	Allows querying for a PostScript device (PS).
.IM	Changed R.1	Up to 14 tokens can be passed to the imbedded file. The tokens are not reset when a macro is called. &0 contains the number of tokens passed.
	Changed R.2	An external file name can be specified in parentheses.
.IN	Changed V/3	An indent can be specified as an increment to or a decrement from the current value.
	Changed R.2	The duration and extent of the indentation can be specified, with or without causing a break.
.IR	New R.1	Specifies indentation from the right-hand edge of the column.
	Changed R.2	The duration and extent of the indentation can be specified, with or without causing a break.
.IS	New R.3	Inserts a specified amount of horizontal white space into the current line.
.IT	New R.1	Enables tracing of symbol substitution, macro substitution, and control word execution.
	Changed R.2	New parameter enables tracing of GML substitution.

Table 14 (Page 5 of 9). Changes to SCRIPT/VS Control Words

Control Word	Code	Description
.IX	New R.2	Causes the index built from .PI control words to be formatted.
	Changed R.3.1	Can specify NOPAGE to prevent page ejects before or after the index.
.JU	New R.3	ON and OFF parameters restore and suspend justification.
.KP	New V/3	Specifies text that is to be kept together and treated as a single block during column balancing.
	Changed R.1	New parameter: INLINE allows you to keep text with preceding and following text. You can specify .KP n, to keep the following n lines together as an INLINE keep. Formatting environment saved around keep.
.LB	New R.1	Executes whenever a line beginning with a blank is processed.
.LI	Changed V/3	Accepts a data line as a parameter. Operands ON and OFF establish and suspend literal interpretation.
.LL	Changed V/3	The line length can be specified as an increment to or a decrement from the current value.
	Changed R.1	Not allowed within a keep.
	Changed R.2	Default is based on the logical device. The control word always takes effect on the next page. When the column width is [implicitly] changed by .LL, the change is reflected in the layout of the current page.
.LO	New R.3	Identifies a place in a column where up to a specified amount of extra vertical white space can be placed during vertical justification.
.LS	New R.3	Controls the depth of lines of text and the expansion and compression ranges for vertical justification.
.LT	New R.1	Executes whenever a line beginning with a tab is processed.
.LY	New R.1	Enables searching of external libraries for symbol and macro definitions.
.ME	New R.2	Ends a macro and returns to the macro's caller. If text or a control word is specified, it is processed as if it were part of the calling file or macro.
.MG	New R.1	Issues user-defined messages.
.NF	New R.3	Cancels or restores concatenation of input lines and justification of output lines and controls whether lines can be allowed to extend beyond the column boundary.
.NL	New R.1	Executes whenever a null input line is processed.
.OC	New R.1	Permits user-defined lines to be inserted into formatted output.
	Changed R.2	User-defined strings can be inserted into formatted lines.
	Changed R.3	Ignored for page printers.
	Changed R.3.2	Allows PostScript comments and AFP page printer comments.
.OF	Changed V/3	An offset can be specified as an increment to or a decrement from the current value. Any new .OF [Offset] control word resets the previous offset value.
.OI	New R.3.2	Inserts overlays and allows them to be included in a named separation master.
.OR	New R.2	Used in conjunction with .IF and .AN to conditionally process text and control words.
	Changed R.3.2	Allows querying for a PostScript device (PS).

Table 14 (Page 6 of 9). Changes to SCRIPT/VS Control Words		
Control Word	Code	Description
.PA	Changed R.1	New parameters: NOSTART ends the current page without starting the new page. You can then modify the page layout, headings, footings, and so on. The next page is not started until SCRIPT/VS encounters either a control word that requires it or input text. ODD, EVEN, ON, and OFF, allow you to page eject to an odd- or even-numbered page.
	Changed R.3.2	Page numbering limit increased to 99,999,999.
.PF	New R.1	Restores the most recently saved font.
.PI	New R.2	Used to build an index. The .IX control word is used to format the index.
.PL	Changed V/3	The page length can be specified as an increment to or a decrement from the current value.
	Changed R.2	The default is based on the logical device. The control word always takes effect on the next page.
.PM	New R.2	Specifies the binding margin and overrides the BIND option of the SCRIPT command.
	Changed R.3	The NOBIND parameter allows the margins to be changed only if the BIND option of the SCRIPT command was not specified.
.PN	Changed V/3	New parameters: FRAC initiates fractional page numbering (decimal point pages). NORM restores normal (ascending integer) page numbering and causes a page eject. PREF specifies a character string prefix for all page numbers. ALPH allows you to specify alphabetic page numbering. n allows you to reset the page number. The control word always takes effect on the next page.
	Changed R.3.2	Page numbering limit increased to 99,999,999.
.PO	New R.3.2	Specifies that an ASCII file of PostScript commands is to be included in a document.
	Changed R.4	New parameter (SEPMASTR) allows you to specify that the PostScript image is to be included in a separation master.
.PT	New V/3	Creates table of contents entries.
	Changed R.1	Both control words and text can be written to the table of contents utility file.
.PW	New R.3	Specifies the width of the logical page, including the page margin (binding).
.PX	New R.3	Specifies that records read from input files that begin with specific control characters be prefixed with given control character strings.
.RC	Changed R.2	The position of the revision code in the intercolumn gutter can be changed. Revision codes are now applied to skip and space, but not to running headings or footings.
	Changed R.3	Revision codes no longer apply to skips (.SK [Skip]) and spaces (.SP [Space]).
	Changed R.3.1	Can now specify revision codes to print LEFT, RIGHT, INSIDE, or OUTSIDE.
	Changed R.4	Allows fonts to be defined for the revision code character. Also allows revision code character to appear to the left of the first column and to the right of the second column when formatting in two-column format.
.RD	Changed R.2	The STOP parameter allows strings to be typed in the middle of formatted column lines.
	Changed R.3	Does not cause a break or section break.
.RE	Changed R.3	Environments can be restored by <i>name</i> .

Table 14 (Page 7 of 9). Changes to SCRIPT/VS Control Words

Control Word	Code	Description
.RF	New R.1	Specifies text and control words to be formatted and placed at the bottom of every page.
	Changed R.2	Running footings can be suppressed and restored without being redefined.
	Changed R.3	Most restrictions on the content of running footings have been removed.
.RH	New R.1	Specifies text and control words to be formatted and placed at the top of every page.
	Changed R.2	Running headings can be suppressed and restored without being redefined.
	Changed R.3	Most restrictions on the content of running headings have been removed.
.RI	Changed V/3	Accepts a text line as a parameter.
.RN	New R.2	Causes line reference numbers to be printed to the right of the page.
	Changed R.3	Ignored for page printers.
.RV	New V/3	Sets a symbol to a value read from the terminal.
	Changed R.1	A user can type in characters without having to enclose them in quotation marks. You can specify a file as the terminal input file by using the .DD [Define Data File-id] control word.
	Changed R.3	If the terminal input source is mapped to a file with .DD, end-of-file can be detected by testing the existence of the named symbol.
.SA	Changed R.3	Environments can be saved by <i>name</i> .
.SB	New R.3	Specifies that subsequent text is to be placed above or below the usual baseline.
.SD	New R.4	Defines shading for boxes, named areas, and table cells when formatting for AFP printers or PostScript devices.
.SE	Changed V/3	The OFF operand cancels a symbol value.
	Changed R.1	New parameters allow you to locate one character string within another or extract a substring. The value of a symbol can be taken from an external library.
	Changed R.2	A symbol can be assigned a quoted string, which is not examined for inner quotation marks.
.SI	New R.3	Identifies a segment to be included in the document at the current column position.
	Changed R.3.1	Can specify NOBREAK to include the segment in the current output line without causing a break. Can specify MAP to indicate if page segments are to be mapped.
	Changed R.4	New parameter (SEPMASR) allows you to specify if the segment is to be included in a separation master. New parameter (ABSOLUTE) allows you to specify your own width and depth value for reserving space instead of using the information in the segment.
.SK	New V/3	Specifies vertical white space to be discarded if it falls at the top of a column.
	Changed R.1	New parameter: P, specifies page-width skips. If a conditional skip is followed by another .SK or .SP, the longer of the two is used, not the second. .SL governs the size of skip requests expressed in lines, even if A is specified. .SK requests expressed in other space units are not affected by the setting of .SL.
	Changed R.3	Specifies vertical white space to be discarded at the end of a column if it is followed by a .CD control word and if .BC OFF was also specified.
.SM	New R.4	Identifies document items to be contained in a particular separation master.

Table 14 (Page 8 of 9). Changes to SCRIPT/VS Control Words		
Control Word	Code	Description
.SO	New R.2	Sets the STAIRS/VS paragraph ID, document name, and user number.
.SP	Changed V/3	New parameters: A specifies absolute spacing. C specifies conditional spacing.
	Changed R.1	New parameter: P specifies page-width space. If a conditional space is followed by another .SP or .SK, the longer of the two is used, not the second. .SL governs the size of space requests expressed in lines, even if A is specified. .SP requests expressed in other space units are not affected by the setting of .SL.
.SU	Changed V/3	Initial value is ON.
	Changed R.2	The line form of .SU no longer turns off subsequent substitution.
.SV	New R.1	Enables spelling verification in conjunction with the SPELLCHK option of the SCRIPT command.
	Changed R.2	Initial value is ON.
.SX	New R.1	Specifies a line composed of a left-justified string, a right-justified string, and a string to fill the space between.
	Changed R.2	New parameter allows a centered string between the left and right strings.
.SY	Reserved	This control word is reserved for diagnostic purposes.
.TA	New R.3.1	Starts a table, row, cell, table heading definition, or table footing definition.
	Changed R.3.2	New parameter (BREAK) ends current row, causes the rows that have been formatted to be placed, causes a column or page eject, and returns the storage that has been used for formatting that portion of the table.
.TB	New R.3.2	Defines how tab characters are to be resolved.
.TC	New V/3	Causes the table of contents to be formatted.
	Changed R.1	Rules for n and input line parameters have been changed.
	Changed R.3.1	Can now specify NOPAGE to prevent page ejects before and after the table of contents.
.TD	New R.3.1	Defines a table, row, or cell to use when building tables.
	Changed R.4	New parameter allows shading of cells on AFP printers and PostScript devices.
.TH	New R.2	Executes only if the most recently executed .IF, .AN, or .OR control word was true.
.TI	New R.1	Specifies character translations that are to be performed before any other processing.
.TM	Changed V/3	The top margin can be specified as an increment to or decrement from the current value.
	Changed R.1	Default is based on the logical device. The control word always takes effect on the next page.
	Changed R.3	Default values changed.
.TP	New R.3	Defines how tab characters in text are to be resolved. Allows you to specify that text is to be left-, center-, right-, or character-aligned at each tab position.
.TS	New R.2	Translates single characters to strings.
.TU	New R.2	Specifies the translation to be performed when capitalizing text.
	Changed R.3.2	Can specify a character-to-string translation for converting to uppercase.

Table 14 (Page 9 of 9). Changes to SCRIPT/VS Control Words

Control Word	Code	Description
.UC	New V/3	Underscores and capitalizes text.
	Changed R.1	New parameters: ON and OFF, allow you to capitalize and underscore large blocks of text.
.UD	Changed R.1	The "required blank" is normally not underscored. The required blank defaults to hexadecimal 41.
	Changed R.2	Indicates <i>only</i> whether the blank is to be underscored. By default, blanks are underscored.
	Changed R.3	On page printers, underscoring is achieved by using a <i>named</i> rule beneath text. The rule can be placed above or below the normal baseline.
.UN	Changed V/3	An undent can be specified as an increment to or decrement from the current value.
.UP	New V/3	Capitalizes text.
	Changed R.1	New parameters: ON and OFF allow you to capitalize large blocks of text.
.US	New V/3	Underscores text.
	Changed R.1	New parameters: ON and OFF allow you to underscore large blocks of text.
.UW	New R.2	Executes when unverified words are found during spelling verification.
.VR	New R.3	Places vertical rules in the column.
.VT	New R.3.1	Indicates the place in the formatted document where variable text is to be inserted by a user's postprocessor at a later time and the amount of horizontal space to be saved for the text.
.WF	New R.1	Specifies lines to be written to an external file.
	Changed R.3	The TAG parameter allows you to use a GML end tag to end a write-to-file mode begun with a GML start tag.
.WS	New R.3	Controls the width of word spaces.
.ZZ	Reserved	This control word is reserved for diagnostic purposes.

Compatibility with TSO/FORMAT

The TSO Data Utility program product provides users of the Time Sharing Option (TSO) of OS/VS2 with a FORMAT function. TSO/FORMAT allows TSO users to enter into TSO text data sets the formatting controls that indicate the type of formatting required.

SCRIPT/VS provides TSO/FORMAT users with an easy migration to more powerful formatting. SCRIPT/VS control word syntax is identical for many TSO/FORMAT control words, and SCRIPT/VS control words that are new to TSO/FORMAT users provide many new or enhanced functions.

Creating a TSO/FORMAT-Compatible Environment

SCRIPT/VS provides a symbol and macro facility that allows you to process TSO/FORMAT documents without modifying the documents themselves. Table 15 lists the TSO/FORMAT control words that are not directly supported by SCRIPT/VS. You can define a SCRIPT/VS macro with the name of the TSO/FORMAT control word that executes the equivalent SCRIPT/VS control word.

You can place your macro definitions in a profile to ensure that they are always available when you process TSO/FORMAT documents. For details, refer to the information that describes the PROFILE option in “SCRIPT Command Options” in the *DCF: SCRIPT/VS Language Reference*.

Table 15. Unsupported TSO/FORMAT Control Words. A SCRIPT/VS control word that provides an equivalent function is listed for each TSO/FORMAT control word, except .RP [Repeat Page] and .PI [Put Index]. SCRIPT/VS provides no equivalent of the Reprint function.	
TSO/FORMAT Control Word	Equivalent SCRIPT/VS Control Word
.AD [Adjust] .BL [Blank] .EN [End] .FL [Float] .HI [Hanging Indent] .NJ [No Justify] .PN [Page Numbering Mode] .RT [Running Title] .ST [Stop]	.RI [Right Adjust] .TR [Translate Character] .CE [Center] OFF .FO [Format Mode] ON .UN [Undent], .OF [Offset] .FO [Format Mode] LEFT .QU [Quit]

Appendix C. IBM 3800 Printing Subsystem Model 1 Fonts Provided with SCRIPT/VS

Appendix C. IBM 3800 Printing Subsystem Model 1 Fonts Provided with	
SCRIPT/VS	247
The SCRIPT/VS Fonts	247
SCRIPT/VS Font Modification	249
Creating a World Trade Font	250
Fonts Available for Use with SCRIPT/VS	252

Appendix C. IBM 3800 Printing Subsystem Model 1 Fonts Provided with SCRIPT/VS

Sixteen IBM 3800 Model 1 fonts are supplied as part of the Document Composition Facility (DCF) for use with the SCRIPT/VS formatter. Each font is a full uppercase and lowercase character set occupying two of the IBM 3800's Writable Character Generation Module (WCGM) buffers.

The SCRIPT/VS fonts are:

- GB10 - Gothic Bold 10-pitch
- GB12 - Gothic Bold 12-pitch
- GI12 - Gothic Italic 12-pitch
- GP12 - Gothic Proportional 12-pitch
- GR10 - Gothic Reverse 10-pitch
- GT10 - Gothic Text 10-pitch
- GT12 - Gothic Text 12-pitch
- GT15 - Gothic Text 15-pitch
- RT10 - Roman Text 10-pitch
- SB12 - Serif Bold 12-pitch
- SI10 - Serif Italic 10-pitch
- SI12 - Serif Italic 12-pitch
- S012 - Serif Overstruck 12-pitch
- ST10 - Serif Text 10-pitch
- ST12 - Serif Text 12-pitch
- ST15 - Serif Text 15-pitch

The SCRIPT/VS Fonts

Each SCRIPT/VS font is composed of four members of SYS1.IMAGELIB.¹¹ For a font named abcd, these members are named:

- XTB1abcd - Character arrangement table for abcd
- GRAFabdA - GCM containing uppercase characters
- GRAFabdB - GCM containing lowercase characters
- GRAFabdC - GCM containing World Trade characters

The contents of the uppercase and lowercase graphic character modification modules (GCMs) are mapped in Figure 7 on page 248.¹²

¹¹ In VSE each SCRIPT/VS font is composed of four phases in the core image library.

¹² Refer to the example section for detailed instructions on creating a world trade font.

Dec Hex	EBCDIC	GCM &GRAF.abdB	EBCDIC	GCM &GRAF.abdB
1 00	40	Real Blank	40	Real Blank
2 01	C1	Uppercase 'A'	81	Lowercase 'A'
3 02	C2	Uppercase 'B'	82	Lowercase 'B'
4 03	C3	Uppercase 'C'	83	Lowercase 'C'
5 04	C4	Uppercase 'D'	84	Lowercase 'D'
6 05	C5	Uppercase 'E'	85	Lowercase 'E'
7 06	C6	Uppercase 'F'	86	Lowercase 'F'
8 07	C7	Uppercase 'G'	87	Lowercase 'G'
9 08	C8	Uppercase 'H'	88	Lowercase 'H'
10 09	C9	Uppercase 'I'	89	Lowercase 'I'
11 0A	4A	Cent Sign (¢)	AD	Open Square Bracket
12 0B	4B	Period (.)	AF	Bullet
13 0C	4C	Less Than (<)	8C	Less Than Or Equal
14 0D	4D	Left Parenthesis ((8B	Opening Brace
15 0E	4E	Plus Sign (+)	9E	Plus Or Minus
16 0F	4F	Logical OR ()	9F	Histogram
17 10	50	Ampersand (&)	13	15-pitch Psuedo-Blank
18 11	D1	Uppercase 'J'	91	Lowercase 'J'
19 12	D2	Uppercase 'K'	92	Lowercase 'K'
20 13	D3	Uppercase 'L'	93	Lowercase 'L'
21 14	D4	Uppercase 'M'	94	Lowercase 'M'
22 15	D5	Uppercase 'N'	95	Lowercase 'N'
23 16	D6	Uppercase 'O'	96	Lowercase 'O'
24 17	D7	Uppercase 'P'	97	Lowercase 'P'
25 18	D8	Uppercase 'Q'	98	Lowercase 'Q'
26 19	D9	Uppercase 'R'	99	Lowercase 'R'
27 1A	5A	Exclamation Point (!)	8F	Cross
28 1B	5B	Dollar Sign (\$)	BD	Close Square Bracket
29 1C	5C	Asterisk (*)	BF	Extended Dash
30 1D	5D	Right Parenthesis ())	9B	Closing Brace
31 1E	5E	Semicolon (;)	11	10-pitch Psuedo-Blank
32 1F	5F	Logical Not (~)	DC	Section Sign
33 20	60	Minus Sign (-)	12	12-pitch Psuedo-Blank
34 21	61	Slash (/)	BE	Not Equal
35 22	E2	Uppercase 'S'	A2	Lowercase 'S'
36 23	E3	Uppercase 'T'	A3	Lowercase 'T'
37 24	E4	Uppercase 'U'	A4	Lowercase 'U'
38 25	E5	Uppercase 'V'	A5	Lowercase 'V'
39 26	E6	Uppercase 'W'	A6	Lowercase 'W'
40 27	E7	Uppercase 'X'	A7	Lowercase 'X'
41 28	E8	Uppercase 'Y'	A8	Lowercase 'Y'
42 29	E9	Uppercase 'Z'	A9	Lowercase 'Z'
43 2A	BB	Lower Right Corner	AB	Lower Left Corner
44 2B	6B	Comma (,)	BC	Upper Right Corner
45 2C	6C	Percent Sign (%)	AC	Upper Left Corner
46 2D	6D	Underscore (_)	6D	Underscore (_)
47 2E	6E	Greater Than (>)	AE	Greater Than Or Equal
48 2F	6F	Question Mark (?)	E0	Reverse Slant
49 30	F0	Digit '0'	B0	Superscript '0'

Figure 7 (Part 1 of 2). SCRIPT/VS 3800 Font GCM Contents

Dec	Hex	EBCDIC	GCM &GRAF.abdA	EBCDIC	GCM &GRAF.abdB
50	31	F1	Digit '1'	B1	Superscript '1'
51	32	F2	Digit '2'	B2	Superscript '2'
52	33	F3	Digit '3'	B3	Superscript '3'
53	34	F4	Digit '4'	B4	Superscript '4'
54	35	F5	Digit '5'	B5	Superscript '5'
55	36	F6	Digit '6'	B6	Superscript '6'
56	37	F7	Digit '7'	B7	Superscript '7'
57	38	F8	Digit '8'	B8	Superscript '8'
58	39	F9	Digit '9'	B9	Superscript '9'
59	3A	7A	Colon (:)	EB	Left Intersection
60	3B	7B	Number Sign (#)	EC	Right Intersection
61	3C	7C	At Sign (@)	CB	Bottom Intersection
62	3D	7D	Apostrophe (')	CC	Top Intersection
63	3E	7E	Equal Sign (=)	FA	Vertical Bar
64	3F	7F	Quotation Mark (")		

Figure 7 (Part 2 of 2). SCRIPT/VS 3800 Font GCM Contents. The character raster images for each SCRIPT/VS font named "abcd" are in Graphic Character Modification modules (GCMs) named GRAFabdA and GRAFabdB.

All characters in all SCRIPT/VS fonts are loaded using the GCM mechanism; none of the IBM-supplied character sets is used. This significantly reduces printer setup time.

Each font's translate table maps the EBCDIC character code into WCGM locations that correspond one-for-one with each character's GCM segment number. For a font named abcd, GCM GRAFabdA occupies the first of two WCGMs, and GRAFabdB occupies the second of two WCGMs.

Note that in each WCGM, location X'00' (segment 1) is a blank and location X'2D' (segment 46) is an underscore. These characters are duplicated because, for purposes of merging characters from overstruck records, the IBM 3800 uses each character's WCGM location, rather than its EBCDIC code, to determine which characters are blanks and which are underscores. (Refer to the *IBM 3800 Printing Subsystem Programmer's Guide*.)

Note also that WCGM location X'3F' (segment 64) is unused in the second of the two WCGMs. This is because WCGM code X'FF' is reserved by the IBM 3800 to signify an unassigned character in the translate table. When a font is loaded into the last two WCGMs rather than the first two, location X'3F' in the second WCGM is relocated to WCGM code X'FF'.

SCRIPT/VS Font Modification

Two distinct steps are required to create a new font or to modify an existing font:

- A new graphic character modification module (GCM) must be created that contains the raster images of the new characters.
- A new character arrangement table must be created that identifies the new GCM and maps each character in the GCM into a WCGM location.

The IEBIMAGE utility program, which is described in the *IBM 3800 Printing Subsystem Programmer's Guide*, must be used to perform these steps.

Creating a World Trade Font

World trade fonts are not supplied with DCF. World trade fonts for the 3800 Printing Subsystem Model 1 are created by using the IEBIMAGE utility to modify an existing font. Source image files containing the raster patterns for world trade characters are provided with DCF and can be used to modify existing graphic character modules (GCMs) and character arrangement tables.

For this example, the GCMs and character arrangement table for the GT10 font will be used as a basis for a world trade font called GTWT. The GTWT font will be identical to GT10, except that the dollar sign will be replaced with the Peseta, the number sign will be replaced with the Pound Sterling sign, and the logical-not sign will be replaced with the Yen sign.

In TSO, the GT10 font consists of three members of the 'SYS1.IMAGELIB' data set:

- XTB1GT10 - Character arrangement table for GT10
- GRAFGTOA - GCM containing uppercase GT10 characters
- GRAFGTOB - GCM containing lowercase GT10 characters

These modules have been generated by running IEBIMAGE with the GT10 source image files supplied with DCF. The IMAGE files contain the GRAPHIC statements necessary to create the uppercase and lowercase GCMs and the TABLE statement necessary to produce the character arrangement table.

A GT10WT source image file is also provided with DCF. This file contains raster patterns for the World Trade characters corresponding to the GT10 font. When IEBIMAGE is run using GT10WT, a GRAFGTOC module is produced in SYS1.IMAGELIB. Figure 8 on page 251 provides the contents of the world trade GCM.

The GT10WT source image file does not contain a TABLE statement. The new character arrangement table must be created from XTB1GT10.

After the modules XTB1GT10, GRAFGTOA, GRAFGTOB, and GRAFGTOC have been generated, IEBIMAGE must be run using the following example statements. These statements will copy and modify XTB1GT10 and GRAFGTOA, and they will include GRAFGTOB to create the new font GTWT:

```
@I INCLUDE GT0A,DELSEG=(28,32,60)
@G GRAPHIC REF=((7,5B),(9,7B),(13,6A)),GCM=GT0C
@N NAME GT0D(R)
@I INCLUDE GT10
@T TABLE GCMLIST=(GT0D,GT0B),LOC=((5F),(6A,1F,0))
@N NAME GTWT(R)
```

The first INCLUDE statement specifies that GTOA, the uppercase GT10 GCM, will be used as the basis for modification. The DELSEG parameter indicates that segment numbers 28, 32, and 60, corresponding to the dollar, logical-not, and number signs in the GCM in Figure 7 on page 248, will be deleted in the new GCM.

The GRAPHIC statement contains a REF and a GCM parameter. The REF parameter specifies the decimal segment positions of characters in the GTOC GCM and the new EBCDIC codes assigned to the characters. Recommended EBCDIC designations are given in Figure 7 on page 248; however, EBCDIC designations are arbitrary, and any unused EBCDIC code may be used. In the above

Segment		EBCDIC	GCM &GRAF.abdC
Dec	Hex		
1	00	5B	Uppercase U Umlaut
2	01	7B	Uppercase A Umlaut
3	02	7C	Uppercase O Umlaut
4	03	5B,E0	Uppercase C Cedilla
5	04	7B	Uppercase Digraph
6	05	7C	Uppercase O Scandinavian
7	06	5B	Peseta
8	07	7B	Uppercase N Tilde
9	08	7B	Pound Sterling
10	09	7B,7C	Uppercase O Tilde
11	0A	7B,7C	Uppercase A Tilde
12	0B	5B	Uppercase A Scandinavian
13	0C	5B	Yen
14	0D	E0,4A	Uppercase E Acute
15	0E	A1,D0	Lowercase U Umlaut
16	0F	C0	Lowercase A Umlaut
17	10	7C	Lowercase O Umlaut
18	11	A1,6A,E0	Lowercase C Cedilla
19	12	C0	Lowercase Digraph
20	13	6A	Lowercase O Scandinavian
21	14	6A	Lowercase N Tilde
22	15	C0,6A	Lowercase O Tilde
23	16	C0,79	Lowercase A Tilde
24	17	D0	Lowercase A Scandinavian
25	18	79,D0,C0,5A	Lowercase E Acute
26	19	A1	Lowercase Ess Tset
27	1A	C0,7C	Lowercase A Grave
28	1B	D0	Lowercase E Grave
29	1C	A1	Lowercase I Grave
30	1D	6A	Lowercase O Grave
31	1E	6A,79	Lowercase U Grave

Figure 8. SCRIPT/VS 3800 Font World Trade GCM Contents. Each SCRIPT/VS font named "abcd" includes a Graphic Character Modification module (GCM) named GRAFabdC, which contains the World Trade National Use Graphics. Their recommended EBCDIC assignments are as shown.

example, (7,5B) indicates that the Peseta sign at segment position 7 in GTOC will be assigned EBCDIC code X'5B'. The Pound Sterling sign at segment 9 will be designated as X'7B'. The Yen sign at position '13' will be designated as X'6A'.

The GCM parameter identifies GTOC as the GCM containing the character segments identified by the REF parameter.

The first NAME statement specifies that the modified GCM containing the original GT10 uppercase characters and the three new replacement World Trade characters will be named GTOD. GTOD will be prefixed by GRAF and will be placed in SYS1.IMAGELIB as GRAFGTOD. The (R) parameter indicates that references to the original copy of the named module (if any) will be deleted.

The second INCLUDE statement is associated with the TABLE statement and indicates that a copy of the GT10 character arrangement table will be used as the basis for the new table.

The TABLE statement contains the GCMLIST and LOC parameters. GCMLIST names GCMs that are to be associated with the new character arrangement table. In this case, the modified uppercase GCM, GTOD, will be included as well as the unmodified GCM, GTOB.

The EBCDIC code for the dollar sign was formerly X'5F'. Because this EBCDIC code will not be used and is being replaced with another EBCDIC code (X'6F'), the X'5F' designation must be removed from the character arrangement table. This is accomplished with the first 'LOC' entry of (5F). The second LOC entry of (6A,1F,0) indicates that the Yen sign scan pattern corresponding to the 8-bit EBCDIC code of 6A (XLOC of 6A in the translate table) will be loaded from GRAFGT10 into character set position X'1F' (the 32nd character position, vacated by the logical not sign) in the first WCGM (set number 0). LOC entries are not needed for the Peseta and Yen signs, because the correspondence between the EBCDIC code and the character position in the WCGM have not been changed. X'5B' is still associated with position X'1B', even though X'1B' will now be loaded with the scan pattern for the Peseta rather than the dollar sign. EBCDIC X'7B' is still associated with position X'3B', but it will be loaded with the scan pattern for the Pound Sterling sign rather than the number sign.

The last NAME statement gives the name of the new character arrangement table, GTWT. GTWT will be prefixed by XTB1 and will be placed in SYS1.IMAGELIB as XTB1GTWT.

Fonts Available for Use with SCRIPT/VS

The fonts illustrated in this appendix are provided with SCRIPT/VS for use with the 3800 Printing Subsystem Model 1 and the 3800 Printing Subsystem Model 3 (only in Compatibility Mode). One or two font names can be specified with the CHARS option of the SCRIPT command (for details, see the section that describes the CHARS option in "SCRIPT/VS Command Options" in the *DCF: SCRIPT/VS Language Reference*). The SCRIPT/VS fonts cannot, in general, be combined in the CHARS option with the IBM 3800 fonts listed in the table titled "Fonts Supplied with the 3800 Printing Subsystem Model 1 and Model 3 Operating in Compatibility Mode (MVS only)" in the *DCF: SCRIPT/VS Language Reference*.

Table 16 lists the fonts provided by SCRIPT/VS for use with the 3800 Printing Subsystem. Each font is a complete set of special, uppercase, and lowercase characters. Any two of these fonts can be specified with the CHARS option of the SCRIPT command.

Table 16. Complete List of Fonts Provided with SCRIPT/VS for the 3800 Printing Subsystem (only for MVS) Fonts Provided with SCRIPT/VS		
Text Fonts	Highlight Fonts	Special Fonts
GT10 Gothic (10-pitch) GT12 Gothic (12-pitch) GT15 Gothic (15-pitch)	GB10 Gothic Bold GB12 Gothic Bold GI12 Gothic Italic	GR10 Gothic Reverse GP12 Proportional
ST10 Serif (10-pitch) ST12 Serif (12-pitch) ST15 Serif (15-pitch)	SI10 Serif Italic SI12 Serif Italic SB12 Serif Bold	RT10 Roman Text SO12 Serif Overstruck

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00																	0F
10																	1F
20																	2F
30																	3F
40																	4F
50																	5F
60																	6F
70																	7F
80																	8F
90																	9F
A0																	AF
B0																	BF
C0																	CF
D0																	DF
E0																	EF
F0																	FF

SCRIPT/VS FONT: GT10

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
40																	4F
50																	5F
60																	6F
70																	7F
80																	8F
90																	9F
A0																	AF
B0																	BF
C0																	CF
D0																	DF
E0																	EF
F0																	FF

SCRIPT/VS FONT: GT12

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00																	0F
10																	1F
20																	2F
30																	3F
40																	4F
50																	5F
60																	6F
70																	7F
80																	8F
90																	9F
A0																	AF
B0																	BF
C0																	CF
D0																	DF
E0																	EF
F0																	FF

SCRIPT/VS FONT: GT15

	0	1	2	3	4	5	
--	---	---	---	---	---	---	--

Figure 9. SCRIPT/VS Fonts: Gothic Text

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00																	0F
10																	1F
20																	2F
30																	3F
40																	4F
50	&																5F
60	- /																6F
70																	7F
80		a	b	c	d	e	f	g	h	i							8F
90		j	k	l	m	n	o	p	q	r							9F
A0			s	t	u	v	w	x	y	z							AF
B0	0	1	2	3	4	5	6	7	8	9							BF
C0		A	B	C	D	E	F	G	H	I							CF
D0		J	K	L	M	N	O	P	Q	R							DF
E0	\		S	T	U	V	W	X	Y	Z							EF
F0	0	1	2	3	4	5	6	7	8	9							FF

SCRIPT/VS FONT: GB10

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
40																	4F
50	&																5F
60	- /																6F
70																	7F
80		a	b	c	d	e	f	g	h	i							8F
90		j	k	l	m	n	o	p	q	r							9F
A0			s	t	u	v	w	x	y	z							AF
B0	0	1	2	3	4	5	6	7	8	9							BF
C0		A	B	C	D	E	F	G	H	I							CF
D0		J	K	L	M	N	O	P	Q	R							DF
E0	\		S	T	U	V	W	X	Y	Z							EF
F0	0	1	2	3	4	5	6	7	8	9							FF

SCRIPT/VS FONT: GB12

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00																	0F
10																	1F
20																	2F
30																	3F
40																	4F
50	&																5F
60	- /																6F
70																	7F
80		a	b	c	d	e	f	g	h	i							8F
90		j	k	l	m	n	o	p	q	r							9F
A0			s	t	u	v	w	x	y	z							AF
B0	0	1	2	3	4	5	6	7	8	9							BF
C0		A	B	C	D	E	F	G	H	I							CF
D0		J	K	L	M	N	O	P	Q	R							DF
E0	\		S	T	U	V	W	X	Y	Z							EF
F0	0	1	2	3	4	5	6	7	8	9							FF

SCRIPT/VS FONT: GI12

Figure 11. SCRIPT/VS Fonts: Gothic Highlight

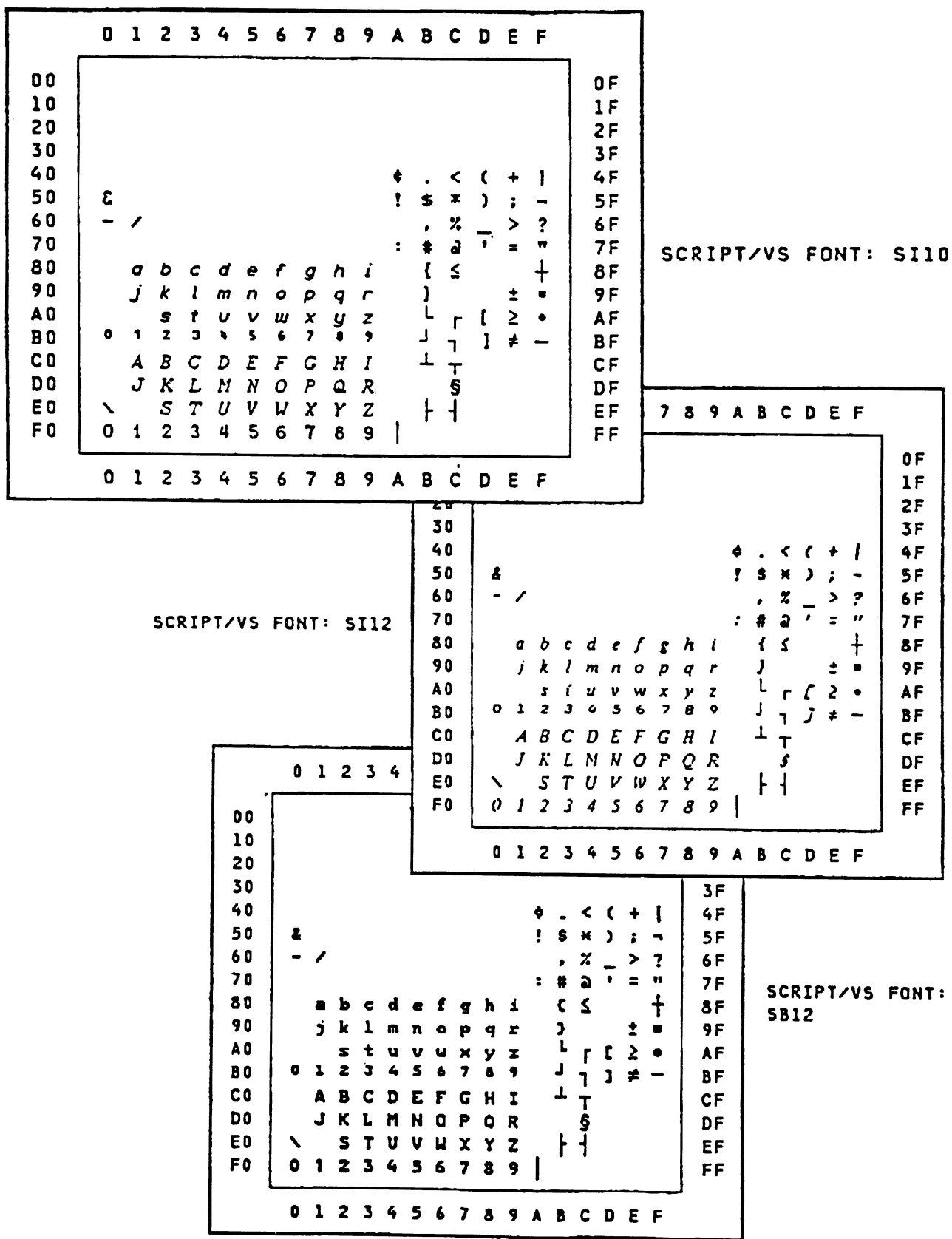


Figure 12. SCRIPT/VS Fonts: Serif Highlight

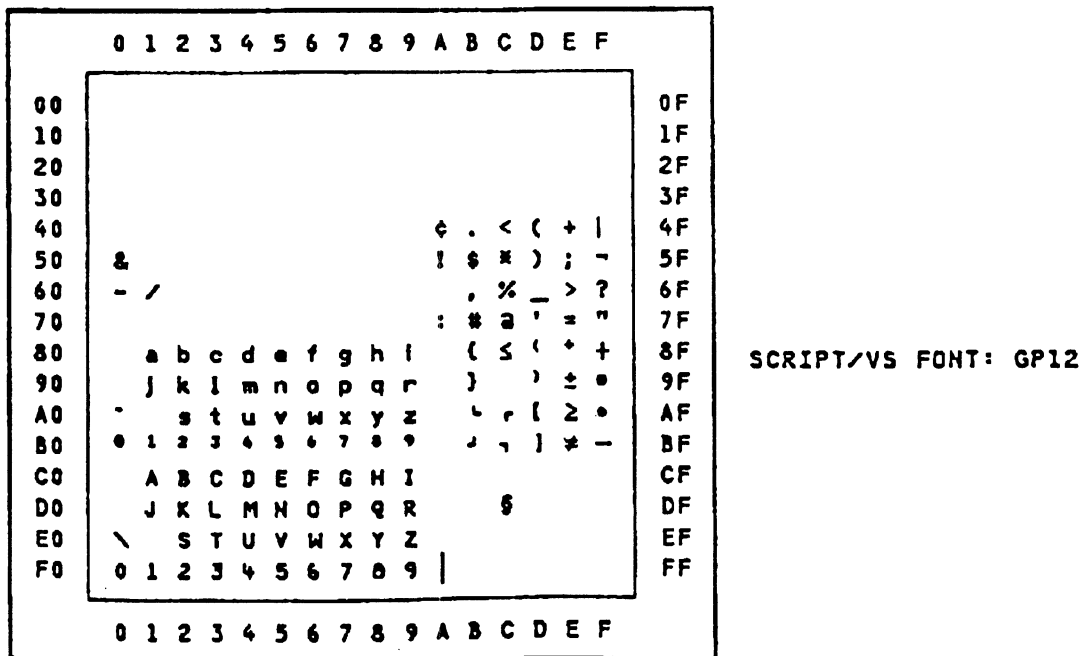
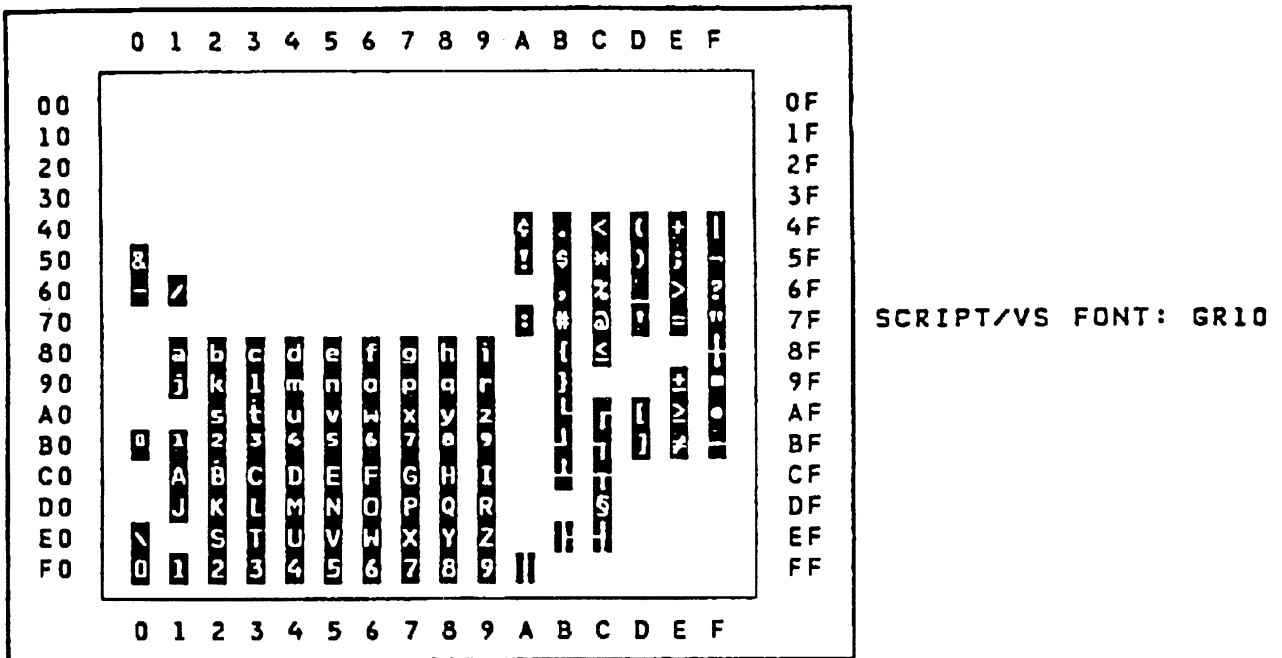


Figure 13. SCRIPT/VS Fonts: Gothic Special Purpose

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00																	0F
10																	1F
20																	2F
30																	3F
40																	4F
50																	5F
60																	6F
70																	7F
80																	8F
90																	9F
A0																	AF
B0																	BF
C0																	CF
D0																	DF
E0																	EF
F0																	FF

SCRIPT/VS FONT:
RT10

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00																	0F
10																	1F
20																	2F
30																	3F
40																	4F
50																	5F
60																	6F
70																	7F
80																	8F
90																	9F
A0																	AF
B0																	BF
C0																	CF
D0																	DF
E0																	EF
F0																	FF

SCRIPT/VS FONT: S012

Figure 14. SCRIPT/VS Fonts: Serif Special Purpose

Appendix D. The DCF Font Library Index Program

Appendix D. The DCF Font Library Index Program	261
Operating Environments	261
CMS	262
Using the Font Library Index Program with TSO	263
Concatenation Considerations with FLIP	263
MVS	264
VSE	265
Font Library Index Program Return Codes	266
Font Library Index Program Report	266

Appendix D. The DCF Font Library Index Program

The Font Library Index Program (FLIP) applies only to IBM page printer typographic fonts. It is used to create an index of the fonts resident in a font library that is needed by SCRIPT/VS for use in determining, from the description of a font given with the .DF [Define Font] control word, whether a requested font is available for the current logical device and text orientation, and if so, what its external (host system) identifier is.

FLIP creates an additional member, named DCFINDEX, for a font library containing these fonts. The DCFINDEX member will contain one structured field, described in Table 17 on page 262, for each set of font objects in the font library that belong to a common typeface family.

The Font Library Index Program also produces a report, illustrated in Figure 15 on page 267 and Figure 16 on page 268, for the 4250 printer. The report lists the contents of the library in the terms that are used when identifying the font with the TYPE parameter of the .DF [Define Font] control word. The coded fonts and code pages are listed in addition to the fonts. This part of the report is shown in Figure 16 on page 268. Figure 17 on page 269 and Figure 18 on page 270 illustrate similar output as it is produced for the 3800 Printing Subsystem Model 3 and for the 3820 Page Printer, respectively.

The listing can then be used to discover what values are valid for the CHARS command option and the TYPE, FONT, and CODEPAGE parameters of the .DF [Define Font] control word, applicable to the font library. The coded font identifier can be used as a value of the CHARS command option and the FONT parameter of the .DF [Define Font] control word. The code page identifier can be used as a value of the CODEPAGE parameter of the .DF [Define Font] control word. The font object identifier is not directly used as a parameter, but it indicates the object used when its descriptor values are specified with the .DF [Define Font] control word TYPE parameter.

The listing can be output to a file or SYSOUT class, depending on the system environment.

Note: FLIP must be used to re-create the DCFINDEX whenever a font library font object is modified. Do *not* put PostScript font resources in the same library with IBM page printer fonts.

See the "SCRIPT/VS System Symbol Names" tables in the *DCF: SCRIPT/VS Language Reference* for a list of the SCRIPT/VS system symbols that return font information.

Operating Environments

For the page printers, the DCF Font Library Index Program will operate in all environments in which font libraries can exist: CMS, MVS, and VSE.

Table 17. Content of a Font Library Index Structured Field. The Font Library Index Program will create one additional member in a font library containing an index based on the attributes of the fonts in the library. Each font in the library will be described in the index in terms of the attributes listed above. The values of the attributes are obtained from various structured fields of each font object.

Field Name	Length	Data Content
SFI	8	Structured Field Introducer
Pattern-Technology-ID	1	Device code (4250, 3800-3, or 3820)
ORIENTATION	4	Inline Direction and Character Rotation (only for the 3800-3)
TYPEFACE	32	Type Face Family Name
Repeating Groups Length	1	14 (bytes)
Common Data Length	1	39 (bytes)
Font-Character-Set 14 bytes		A repeating group — one per font
POINTS	2	Font point size (in decipoints)
WEIGHT	1	Weight class code (0-9)
WIDTH	1	Width class code (0-9)
ATTRIBUTES	2	Font Flags (ITALIC, OUTLINE, and so on)
HOST NAME	8	Font Object's external (host system) name

CMS

If FLIP is invoked without any parameters, a filetype of FONT3820 and a filemode of A is assumed.

In CMS, a font library is a collection of CMS files, with a common default or user-specified file type, residing on a CMS disk or its extensions; font objects are individual CMS files, and the font library index is a CMS file with the file name DCFINDEX.

For example, a small library of Monotype Times New Roman fonts might consist of the following files:

```
AFT02007 FONT4250 G1
AFT02008 FONT4250 G1
AFT03007 FONT4250 G1
AFT03008 FONT4250 G1
AFT04007 FONT4250 G1
AFT04008 FONT4250 G1
AFT05007 FONT4250 G1
AFT05008 FONT4250 G1
```

To create the index, the program must examine each file of the library on the disk and its extensions and, if the file is a font object, add an index entry.

To create the font library index, the 'G' disk must be accessed READ/WRITE and the following command issued:

```
dsmcmf40 font4250 g
```

The program DSMCMF40 will create a CMS file named DCFINDEX FONT4250 and a report named FONT4250 LISTING. DCFINDEX FONT4250 will be placed on the G1 disk because only "g" was specified as the filemode. You can request the file be assigned a specific filemode number by specifying it as the filemode. For example,

```
dsmcmf40 font4250 g5
```

will assign the file with a filemode of G5. The report will be created as a LISTING file type on the same disk. In this example, the report will be written in the file FONT4250 LISTING G5.

To improve performance in the CMS environment, you should not have the disk that contains the existing DCFINDEX as your primary READ/WRITE disk. You can set up a temporary disk for the purpose of holding the new DCFINDEX and make the font disks extensions of that disk. This enables others to use the fonts while FLIP is running.

The order of extensions used by FLIP to build the DCFINDEX must be the same for SCRIPT/VS and Print Services Facility (PSF). If SCRIPT/VS does not use the same order as FLIP, formatting results are unpredictable.

Using the Font Library Index Program with TSO

FLIP creates an index (DCFINDEX) of the fonts contained in a font library. SCRIPT/VS uses DCFINDEX to determine whether a requested font is available for the current logical device and text orientation, and if so, what its external identifier is. FLIP also produces a report called the Font Library Index Program Report that lists the contents of the font library.

To improve the usability of FLIP in TSO, FLIP now allows you to concatenate multiple partitioned data sets for the FONTLIB ddname. A new ddname, FONTLIBO, has been added to specify the single partitioned data set that will contain the DCFINDEX.

The 8K restriction on the number of fontlib members has been removed. The maximum number of fontlib members that FLIP can process in TSO is determined by the amount of storage that is available when you use FLIP.

Concatenation Considerations with FLIP

If concatenation is used for FONTLIB, you must use the new FONTLIBO ddname to specify the single partitioned dataset that will contain the DCFINDEX created by FLIP.

If your FONTLIB is not concatenated, FONTLIBO is ignored. The DCFINDEX will be created in the single partitioned data set referred to by FONTLIB.

If FONTLIB is a concatenated data set, only the first occurrence of each member is used. Subsequent (duplicate) members found in FONTLIB are ignored.

When you use concatenation for FONTLIB with FLIP, the same concatenation order must be used with SCRIPT/VS and PSF. If SCRIPT/VS does not use the same concatenation order as FLIP, formatting results are unpredictable.

For example, if you create a DCFINDEX like this:

```
//INDEX    JOB ...
//          EXEC PGM=DSMBOF40
//STEPLIB DD DSN=SCRIPT.R40.DCFLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//FONTLIB DD DISP=OLD,DSN=userid.MY.FONT3820
//          DD DISP=OLD,DSN=SYS1.FONT3820
//FONTLIB0 DD DISP=OLD,DSN=userid.MY.FONT3820
//
```

then the SCRPTFNT ddname must be used with SCRIPT/VS to use the DCFINDEX created above. For example:

```
ALLOCATE DD(SCRPTFNT) DSN(MY.FONT3820 'SYS1.FONT3820') SHR
SCRIPT MY.TEXT DEV(3820A) FILE(MY.LIST3820)
```

When your installation has the ATMS or DLF feature, formatting results are unpredictable if you use the DCFINDEX created by concatenation. Therefore, with the ATMS or DLF feature, we recommend that you do not place the DCFINDEX created by concatenation in a common library such as 'SYS1.FONT3820'.

MVS

In MVS, a font library is a partitioned data set, font objects are partitioned data set members (with a maximum of 8192), and the font library index is a member named DCFINDEX.

For example, a small set of Monotype Times New Roman fonts in SYS1.FONT4250 might contain the following member names:

```
AFT02007
AFT02008
AFT03007
AFT03008
AFT04007
AFT04008
AFT05007
AFT05008
```

To create the index, the Font Library Index Program must examine each member of the library listed in the partitioned data set directory and, if the member is a font object, add an index entry for the font.

The following job must be executed to create the font library index:

```
//INDEX    JOB ...
//          EXEC PGM=DSMBOF40
//STEPLIB DD DSN=SCRIPT.R40.DCFLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=A
//FONTLIB DD DISP=OLD,DSN=SYS1.FONT4250
//
```

The program DSMBOF40 will add one member, named DCFINDEX, to SYS1.FONT4250. The SYSPRINT DD defines where the report will be listed. In the example JCL, the listing is SYSOUT; it may be preserved online by specifying a dataset with RECFM = VBM, BLKSIZE = 1022, and LRECL = 125.

For the 3800 Printing Subsystem Model 3 and the 3820 Page Printer, the only necessary change to the example JCL above would be the data set name of the font library containing 3800 Printing Subsystem Model 3 and 3820 Page Printer fonts. In this particular example, that data set name would be SYS1.FONT38PP for the 3800 Printing Subsystem Model 3 and SYS1.FONT3820 for the 3820 Page Printer.

VSE

In VSE/AF Version 2, fonts are stored in a sublibrary. Font objects are library phases (with a maximum of 8192), and the font library index is a phase named DCFINDEX.

For example, a small set of Monotype Times New Roman fonts in the FONT4250.LIBRARY might contain the following phases:

```
AFT02007
AFT02008
AFT03007
AFT03008
AFT04007
AFT04008
AFT05007
AFT05008
```

To create the index, the FLIP must examine each member of the library listed in the library directory and, if the member contains a font object, create an index entry for the font. The DCFINDEX phase composed of the index is produced by using the VSE linkage editor; DSMBDF40 produces the input for the linkage editor.

In VSE/AF Version 2, the following job must be executed to create the font library index:

```
// JOB DCFINDEX
  LIBDEF PHASE,SEARCH=(DCF.LIB40,DCF.FONTLIB),CATALOG=DCF.FONTLIB
// DLBL IJSYSPH,'DCF.SYSPCH.FILE1',0,SD
// EXTENT ,SYSWK1,,,150,50
ASSGN SYSPCH,3350,VOL=SYSWK1,SHR
// EXEC DSMBDF40,SIZE=AUTO
/*
CLOSE SYSPCH,PUNCH
// DLBL IJSYSIN,'DCF.SYSPCH.FILE1',99/365
// EXTENT SYSIPT,SYSWK1
ASSGN SYSIPT,3350,PERM,VOL=SYSWK1,SHR
// EXEC LIBR,PARM='A S=DCF.FONTLIB'
/*
  LIBDEF OBJ,SEARCH=DCF.FONTLIB
// OPTION CATAL
  PHASE DCFINDEX,*
  INCLUDE DCFINDEX
/*
// EXEC LNKEDT
CLOSE SYSIPT,UA†
/*
/&
```

† UA is an installation-dependent unit address of the SYSIPT before this job is executed. See your system programmer for the correct address.

The sublibrary list specified in the VSE "LIBDEF" statement for the "EXEC DSMBDF40" job step must not contain any empty sublibraries. If an empty sublibrary is specified, DSMBDF40 will terminate with a return code of 32.

Font Library Index Program Return Codes

The Font Library Index Program sets a return code resulting from index processing. Table 18 lists these return codes.

Table 18. Font Library Index Program Return Codes	
Return Code	Meaning
0	Normal completion
4	Listing file OPEN error
8	No font objects in the font library
12	Font library OPEN error
16	Not enough storage available for processing
20	Font library read error
24	Unable to FIND font object in the font library
28	Unable to read or write to disk
32	Librarian macro services error
36	Fontlib data set blocksize too small
40	Font library member name table overflow (MVS and VSE). The input font library is too large to be processed by FLIP. The library will have to be subdivided into two or more smaller font libraries
44	Unable to access disk specified with filemode (VM).

Font Library Index Program Report

The Font Library Index Program report, illustrated in Figure 15 on page 267, Figure 16 on page 268, Figure 17 on page 269, Figure 18 on page 270, Figure 19 on page 271, and Figure 20 on page 272, provides a list of the contents of a font library described as it would be used with the TYPE option of the .DF [Define Font] control word. Note that the typeface name is factored out of each entry.

The fonts for each typeface family and orientation are grouped together. Within each typeface family and orientation group, inline direction and character rotation, fonts are ordered by point size, weight class, width class, and selection flags (attributes). The identity of the font object is given for each entry. Also, for a particular entry, the default line space, figure space, and word space are presented for informational purposes.

Point sizes can range from 6 through 72.

Values for weight, width, and attributes are identified in the description of the TYPE parameter of the .DF [Define Font] control word.

Font objects that are identically described in DCFINDEX will be noted as duplicates.

```

DSMFPRPT: DCF FONT LIBRARY INDEX PROGRAM REPORT.      FONT LIBRARY: FONT4250      PAGE: 41
For a description of this report see the SCRIPT/VS USERS GUIDE chapter on SELECTING FONTS
                                         FORMAT: BOUNDED BOX

TYPEFACE NAME: MONOTYPE ROCKWELL E1
FONT      POINT
IDENTIFIER SIZE  WEIGHT  WIDTH  ATTRIBUTES  DEVICE  LINE  FIGURE WORD
                SPACE  SPACE  SPACE
AFT20159   24   Bold    Normal
AFT20157   24   Bold    Normal    Italic     4250   213
AFT26158   30   Medium  Normal
AFT26156   30   Medium  Normal    Italic     4250   268
AFT26159   30   Bold    Normal
AFT26157   30   Bold    Normal    Italic     4250   268
AFT32158   36   Medium  Normal
AFT32156   36   Medium  Normal    Italic     4250   321
AFT32159   36   Bold    Normal
AFT32157   36   Bold    Normal    Italic     4250   321
AFT44158   48   Medium  Normal
AFT44156   48   Medium  Normal    Italic     4250   428
AFT44159   48   Bold    Normal
AFT44157   48   Bold    Normal    Italic     4250   428
AFT56158   60   Medium  Normal
AFT56156   60   Medium  Normal    Italic     4250   535
AFT56159   60   Bold    Normal
AFT56157   60   Bold    Normal    Italic     4250   535
AFT64158   72   Medium  Normal
AFT64156   72   Medium  Normal    Italic     4250   642
AFT64159   72   Bold    Normal
AFT64157   72   Bold    Normal    Italic     4250   642

TYPEFACE NAME: MONOTYPE TIMES NEW ROMAN
FONT      POINT
IDENTIFIER SIZE  WEIGHT  WIDTH  ATTRIBUTES  DEVICE  LINE  FIGURE WORD
                SPACE  SPACE  SPACE
AFT02008   06   Medium  Normal
AFT02006   06   Medium  Normal    Italic     4250   53   26   17
AFT02009   06   Bold    Normal
AFT02007   06   Bold    Normal    Italic     4250   53   27   17
AFT03008   07   Medium  Normal
AFT03006   07   Medium  Normal    Italic     4250   61   30   19
AFT03009   07   Bold    Normal
AFT03007   07   Bold    Normal    Italic     4250   61   31   20
AFT04008   08   Medium  Normal
AFT04006   08   Medium  Normal    Italic     4250   70   33   21
AFT04009   08   Bold    Normal
AFT04007   08   Bold    Normal    Italic     4250   70   33   21
AFT05008   09   Medium  Normal
AFT05006   09   Medium  Normal    Italic     4250   80   39   25
AFT05009   09   Bold    Normal
AFT05007   09   Bold    Normal    Italic     4250   80   39   25
AFT06008   10   Medium  Normal

```

Figure 15. Sample of Font Library Index Program Listing for the 4250. Other attributes of a font can include UNDERSCORE, OUTLINE, and OVERSTRUCK.

FONT IDENTIFIER: AFT06008 MONOTYPE TIMES NEW ROMAN

CODED FONT	CODE PAGE
AFTTR361	AFTC0361 INTERNATIONAL SET 5 (4250)
AFTTR382	AFTC0382 AUSTRIA, GERMANY, SWITZERLAND
AFTTR383	AFTC0383 BELGIUM
AFTTR384	AFTC0384 BRAZIL
AFTTR385	AFTC0385 CANADA (FRENCH)
AFTTR386	AFTC0386 DENMARK, NORWAY
AFTTR387	AFTC0387 FINLAND, SWEDEN
AFTTR388	AFTC0388 FRANCE, LUXEMBURG, SWITZERLAND
AFTTR389	AFTC0389 ITALY, SWITZERLAND (ITALIAN)
AFTTR390	AFTC0390 JAPAN (LATIN)
AFTTR391	AFTC0391 PORTUGAL
AFTTR392	AFTC0392 SPAIN, PHILIPPINES
AFTTR393	AFTC0393 LATIN AMERICA (SPANISH SPEAKING)
AFTTR394	AFTC0394 UNITED KINGDOM, AUSTRALIA
AFTTR395	AFTC0395 UNITED STATES, CANADA (ENGLISH)

Figure 16. Sample of Font Library Index Program Listing for the 4250. The coded font and code page section.

DSMFPRPT: DCF FONT LIBRARY INDEX PROGRAM REPORT. FONT LIBRARY: FONT38PP PAGE: 2
 For a description of this report see the SCRIPT/VS USERS GUIDE chapter on SELECTING FONTS
 FORMAT: UNBOUNDED BOX

TYPEFACE NAME: EMULATION 5						DIRECTION: 000		ROTATION: 000	
FONT IDENTIFIER	POINT SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE	LINE SPACE	FIGURE SPACE	WORD SPACE	
C1RCDW90	09	Medium	Normal		3800-3	40	48	48	

TYPEFACE NAME: ISIL GOTHIC						DIRECTION: 000		ROTATION: 000	
FONT IDENTIFIER	POINT SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE	LINE SPACE	FIGURE SPACE	WORD SPACE	
C10005ZA	05	Medium	Normal		3800-3	22	9	9	
C10007ZA	07	Medium	Normal		3800-3	29	12	12	
C10007ZC	07	Medium	Normal	Italic	3800-3	29	12	12	
C10007ZB	07	Bold	Normal		3800-3	29	12	12	
C10007ZD	07	Bold	Normal	Italic	3800-3	29	12	12	
C10008ZA	08	Medium	Normal		3800-3	31	13	13	
C10008ZC	08	Medium	Normal	Italic	3800-3	31	13	13	
C10008ZB	08	Bold	Normal		3800-3	31	13	13	
C10008ZD	08	Bold	Normal	Italic	3800-3	31	13	13	
C10009ZA	09	Medium	Normal		3800-3	36	15	15	
C10009ZC	09	Medium	Normal	Italic	3800-3	36	15	15	
C10009ZB	09	Bold	Normal		3800-3	36	15	15	
C10009ZD	09	Bold	Normal	Italic	3800-3	36	15	15	
C10010ZA	10	Medium	Normal		3800-3	38	16	16	
C10010ZC	10	Medium	Normal	Italic	3800-3	38	16	16	
C10010ZB	10	Bold	Normal		3800-3	38	16	16	
C10010ZD	10	Bold	Normal	Italic	3800-3	38	16	16	
C10012ZA	12	Medium	Normal		3800-3	48	20	20	
C10012ZC	12	Medium	Normal	Italic	3800-3	48	20	20	
C10012ZB	12	Bold	Normal		3800-3	48	20	20	
C10012ZD	12	Bold	Normal	Italic	3800-3	48	20	20	

TYPEFACE NAME: ISIL GOTHIC REVERSE						DIRECTION: 000		ROTATION: 000	
FONT IDENTIFIER	POINT SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE	LINE SPACE	FIGURE SPACE	WORD SPACE	
C10007ZE	07	Medium	Normal		3800-3	29	12	12	
C10008ZE	08	Medium	Normal		3800-3	31	13	13	
C10009ZE	09	Medium	Normal		3800-3	36	15	15	
C10010ZE	10	Medium	Normal		3800-3	38	16	16	
C10012ZE	12	Medium	Normal		3800-3	48	20	20	

Figure 17. Sample of Font Library Index Program Listing for the 3800 Printing Subsystem Model 3. Other attributes of a font can include UNDERSCORE, OUTLINE, OVERSTRUCK, AND ORIENTATION.

FONT IDENTIFIER: C1Q07580 PI SERIF

CODED FONT	CODE PAGE
------------	-----------

X1Q0758P	T1GPI363 PI FONTS
----------	-------------------

FONT IDENTIFIER: C1T05500 SONORAN SERIF

CODED FONT	CODE PAGE
------------	-----------

X1T00361	T1GI0361	INTERNATIONAL SET 5
X1T00382	T1GI0382	AUSTRIA, GERMANY, SWITZERLAND
X1T00383	T1GI0383	BELGIUM
X1T00384	T1GI0384	BRAZIL
X1T00385	T1GI0385	CANADA (FRENCH)
X1T00386	T1GI0386	DENMARK/NORWAY
X1T00387	T1GI0387	SWEDEN/FINLAND
X1T00388	T1GI0388	FRANCE, LUXEMBOURG, SWITZERLAND
X1T00389	T1GI0389	ITALY, SWITZERLAND (ITALIAN)
X1T00390	T1GI0390	JAPAN (LATIN)
X1T00391	T1GI0391	PORTUGAL
X1T00392	T1GI0392	SPAIN/PHILIPPINES
X1T00393	T1GI0393	LATIN AMERICA (SPANISH)
X1T00394	T1GI0394	U.K., AUSTRAL., IRE., H.K., N.Z.
X1T00395	T1GI0395	UNITED STATES, CANADA (ENGLISH)

Figure 18. Sample of Font Library Index Program Listing for the 3800 Printing Subsystem Model 3. The coded font and code page section.

DSMFPRPT: DCF FONT LIBRARY INDEX PROGRAM REPORT. FONT LIBRARY: FONT3820 PAGE: 1
 For a description of this report see the SCRIPT/VS USERS GUIDE chapter on SELECTING FONTS
 FORMAT: BOUNDED BOX

TYPEFACE NAME: APL						LINE	FIGURE	WORD
FONT	POINT					SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE			
COS0AE20	05	Medium	Normal		3820	20	12	12
COS0AE10	10	Medium	Normal		3820	40	24	24

TYPEFACE NAME: APL2						LINE	FIGURE	WORD
FONT	POINT					SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE			
COS0AP20	05	Medium	Normal		3820	20	12	12
COL0AG15	08	Medium	Normal		3820	30	16	16
COL0AI12	09	Medium	Normal	Italic	3820	30	20	20
COL0AI10	10	Medium	Normal	Italic	3820	40	24	24

TYPEFACE NAME: APL2 BASE						LINE	FIGURE	WORD
FONT	POINT					SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE			
COL00APL	10	Medium	Normal	Italic	3820	40	24	24

TYPEFACE NAME: APL2 BASE10						LINE	FIGURE	WORD
FONT	POINT					SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE			
COS0AP10	10	Medium	Normal	Italic	3820	40	24	24

TYPEFACE NAME: APL2 BASE13						LINE	FIGURE	WORD
FONT	POINT					SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE			
COS0AP13	08	Medium	Normal	Italic	3820	28	18	18

Figure 19. Sample of Font Library Index Program Listing for the 3820 Page Printer. Other attributes of a font can include UNDERSCORE, OUTLINE, and OVERSTRUCK.

FONT IDENTIFIER: C0A05300 SONORAN SANS SERIF
CODED FONT CODE PAGE
X0AC0361 T1GI0361 INTERNATIONAL SET 5
X0AC0382 T1GI0382 AUSTRIA, GERMANY, SWITZERLAND
X0AC0383 T1GI0383 BELGIUM
X0AC0384 T1GI0384 BRAZIL
X0AC0385 T1GI0385 CANADA (FRENCH)
X0AC0386 T1GI0386 DENMARK/NORWAY
X0AC0387 T1GI0387 SWEDEN/FINLAND
X0AC0388 T1GI0388 FRANCE, LUXEMBOURG, SWITZERLAND
X0AC0389 T1GI0389 ITALY, SWITZERLAND (ITALIAN)
X0AC0390 T1GI0390 JAPAN (LATIN)
X0AC0391 T1GI0391 PORTUGAL
X0AC0392 T1GI0392 SPAIN/PHILIPPINES
X0AC0393 T1GI0393 LATIN AMERICA (SPANISH)
X0AC0394 T1GI0394 U.K., AUSTRAL., IRE., H.K., N.Z.
X0AC0395 T1GI0395 UNITED STATES, CANADA (ENGLISH)

FONT IDENTIFIER: C0L0AD10 (Not Available)
CODED FONT CODE PAGE
X0AD0A T1000293 APL (USA)
X0AD0B T1000310 GRAPHIC ESCAPE APL/TN
X0AD10 T1L0AD10 APL2

FONT IDENTIFIER: C0L0AD12 (Not Available)
CODED FONT CODE PAGE
X0AD12 T1L0AD10 APL2
X0AD2A T1000293 APL (USA)
X0AD2B T1000310 GRAPHIC ESCAPE APL/TN

Figure 20. Sample of Font Library Index Program Listing for the 3820 Page Printer. The coded font and code page section.

DSMFPRPT: DCF FONT LIBRARY INDEX PROGRAM REPORT. FONT LIBRARY: FONT300 PAGE: 1
 For a description of this report see the SCRIPT/VS USERS GUIDE chapter on SELECTING FONTS
 FORMAT: BOUNDED BOX

TYPEFACE NAME: APL2							LINE	FIGURE	WORD
FONT	POINT						SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE				
COLOAG12	09	Medium	Normal		3820		30	20	20
COLOAG10	10	Medium	Normal		3820		40	24	24

TYPEFACE NAME: APL2 TEXT							LINE	FIGURE	WORD
FONT	POINT						SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE				
COLOAT12	09	Medium	Normal		3820		30	20	20

TYPEFACE NAME: BAR CODE							LINE	FIGURE	WORD
FONT	POINT						SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE				
COBAR050	36	Medium	Normal		3820		144	60	24
COBAR100	72	Medium	Normal		3820		288	120	24

TYPEFACE NAME: BOOK							LINE	FIGURE	WORD
FONT	POINT						SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE				
COS0D0TR	10	Medium	Normal		3820		40	20	20

TYPEFACE NAME: TIMES ROMAN							LINE	FIGURE	WORD
FONT	POINT						SPACE	SPACE	SPACE
IDENTIFIER	SIZE	WEIGHT	WIDTH	ATTRIBUTES	DEVICE				
COE0T260	06	Medium	Normal		RELATIVE	6420	3000	1500	
COE0T280	08	Medium	Normal		RELATIVE	8560	4000	2000	
COE0T200	10	Medium	Normal		RELATIVE	10700	5000	2500	
COE0T280	12	Medium	Normal		RELATIVE	12840	6000	3000	
COE0T200	14	Medium	Normal		RELATIVE	14980	7000	3500	
COE0T2H0	18	Medium	Normal		RELATIVE	19260	9000	4500	
COE0T2N0	24	Medium	Normal		RELATIVE	25680	12000	6000	

Figure 21. Sample of Font Library Index Program Report for the IBM LaserPrinter 4028

Appendix E. PostScript Information

Appendix E. PostScript Information	277
PostScript Font Support	277
Adobe Font Metrics Files	277
Adobe Font Metrics Files	278
Space Unit Notation	279
PostScript Code Pages	279
Modifying a PostScript Code Page	280
The PostScript DCFINDEX File	281
Adding DCFINDEX File Entries	284
Customizing DCFINDEX File Entries	285
Configuring Your Printer for 8-Bit ASCII	286
PostScript Output Structuring Conventions	286
PostScript Command Abbreviations	287
Other PostScript Considerations	287

Appendix E. PostScript Information

To support PostScript devices configured to accept 8-bit ASCII, SCRIPT/VS can generate formatted output in PostScript and access PostScript font information in the PostScript font library.

This appendix provides information for the text programmer about PostScript font support, structuring conventions for PostScript output, and PostScript command abbreviations used by DCF. For user information about PostScript, refer to the PostScript appendix in the *DCF: SCRIPT/VS User's Guide*.

PostScript Font Support

DCF provides PostScript font support that is similar to the font support provided for IBM page printers. However, some of the information contained in IBM page printer fonts must be specified for PostScript fonts.

For example, the Logical Device Table (LDT) contains a point size, line space, and code page for the default PostScript font. This information is contained in font files for IBM page printers. The PostScript font values in the LDT can be changed at installation time. See "DSMLPLDT: Logical Device Definitions" on page 131 for a description of how the font values in the LDT can be changed. These defaults can also be changed through SCRIPT control words. Refer to "Using SCRIPT/VS with PostScript Devices" in the *DCF: SCRIPT/VS User's Guide*.

The PostScript font library is set up similar to a font library for a page device. The text programmer can customize this library (only for VM or MVS) by adding, deleting, or modifying PostScript font information. The PostScript font library contains the following objects:

- Adobe Font Metrics files
- PostScript code pages
- A PostScript DCFINDEX file.

These files must be variable length record format.

PostScript fonts must be in their own font library. Do *not* mix PostScript fonts with IBM page printer fonts.

Adobe Font Metrics Files

Adobe Font Metrics (AFM) files provide information that SCRIPT/VS requires to format a document. Forty-three AFM files are shipped with DCF and must be accessible in the PostScript font library.

Important: A font index file, the Adobe Font Metrics files, and the PostScript code page files for the supported fonts are shipped on an "As Is" basis without warranty of any kind either expressed or implied.

To produce PostScript output, a corresponding font file for each AFM file must be available at the PostScript device. For example, to match the fonts used by the unmodified GML Starter Set, both Times Roman and Courier typeface families must be available at the PostScript device.

Adobe Font Metrics Files

The following table shows the Adobe Font Metrics (AFM) files shipped with Release 4:

Typeface Family	AFM Filename
ITC Avant Garde Gothic	PSFAGB, PSFAGBO, PSFAGD, PSFAGDO
ITC Bookman	PSFBKD, PSFBKDI, PSFBKL, PSFBKLI
Courier	PSFCR, PSFCRB, PSFCRBO, PSFCRO
ITC Garamond	PSFGRB, PSFGRBI, PSFGRL, PSFGRLI
Helvetica Black	PSFHVBK, PSFHVBKO
Helvetica Light	PSFHVL, PSFHVLO
Helvetica	PSFHVB, PSFHVBO, PSFHV, PSFHVO
ITC Korinna	PSFKOR, PSFKORB, PSFKORK, PSFKORKB
New Century Schoolbook	PSFNCS, PSFNCSB, PSFNCSBI, PSFNCSI
Palatino	PSFPL, PSFPLB, PSFPLBI, PSFPLI
Times	PSFTR, PSFTRB, PSFTRBI, PSFTRI
Symbol	PSFSYM
ITC Zapf Chancery	PSFZCMI
ITC Zapf Dingbats	PSFZD

Note: The Times Roman and Courier AFM files are shipped with DCF.

To add an AFM file to the PostScript font library on the host system, you must upload the AFM file from an IBM Personal Computer to the host system.

When you upload files, ensure that you do the following:

- Specify an ASCII-to-EBCDIC conversion for each file you send to the host. You should specify Extended ASCII if your uploading program supports the option.
- Specify that carriage return characters or line feed characters are recognized as record separators and are deleted before the file is stored.
- Specify the maximum logical record length and a variable record format. For example, if you upload files using the IBM 3270 Control Program, specify a record length of 255 (Irecl 255).

You can create or modify PostScript fonts and AFM files using procedures described in the *PostScript Language Reference Manual*.¹³ For performance considerations, follow the Adobe AFM interchange format conventions when you create or modify AFM files.

If real numbers are used as key values, values to the right of the decimal point are truncated. No error message is issued. Radix and exponential notation will result in an error message.

¹³ This book is copyrighted by Adobe Systems Incorporated and published by the Addison-Wesley Publishing Company.

Space Unit Notation

Because AFM font information does not contain the necessary em-space, figure space, or interword space, these values are calculated for proportional fonts, using the following algorithm:

1. Calculate the em-space for the font.
 - a. If the AFM file contains the “*emdash*” character, use its width for the width of the em-space.
 - b. If the “*emdash*” character was not found in the AFM file, use two times the width of the “*endash*” character if there is one.
 - c. If neither the “*emdash*” nor the “*endash*” characters are found in the AFM file, use four times the width of the “*space*” character if there is one.
 - d. Otherwise, use the width of the point size.
2. Set the interword space to 1/3 of the calculated em-space.
3. Set the figure space to 1/2 of the calculated em-space.

For nonproportional fonts, the em-space is calculated as detailed above. However, the interword space and the figure space are set to the em-space value.

PostScript Code Pages

PostScript code page files contain information that maps character names to code points. For general information about code pages, refer to “Selecting Fonts” in the *DCF: SCRIPT/VS User’s Guide*.

The code pages for PostScript fonts are based on one of three PostScript character sets:

- Standard
- Symbol
- Dingbat

The following code pages, also shipped with DCF, are based on the code point assignments in the IBM Font licensed programs:

PSCP0361	International Set 5
PSCP0382	German
PSCP0385	Canadian French
PSCP0386	Danish, Norwegian
PSCP0387	Finnish, Swedish
PSCP0388	French
PSCP0389	Italian
PSCP0391	Portuguese
PSCP0392	Spanish
PSCP0394	U. K. English
PSCP0395	American English, Dutch, Canadian English

Note: The default code page, unless modified at your installation, is PSCP0395.

The following Country Extended Code Page files are shipped with DCF. The code point assignments are based on Country Extended Code Pages (CECP) Version 1.

PSGDP037	American English, Dutch, Canadian English, Canadian French
PSGDP273	German
PSGDP277	Danish, Norwegian
PSGDP278	Finnish, Swedish
PSGDP280	Italian
PSGDP282	Portuguese
PSGDP284	Spanish
PSGDP285	U. K. English
PSGDP297	French
PSGDP871	Icelandic

Two additional code pages for PostScript fonts are:

PSCPSYM	Symbol Font
PSCPDIN	Dingbat Font

The data set names for the code page files coincide with the font library naming conventions in the various operating environments:

CMS	PSCPxxxx FONTPS <i>fm</i>
MVS	SCRIPT.R40.FONTPS(PSCPxxxx)
VSE	The sublibrary is FONTPS. The phase name is PSCPxxxx.

Modifying a PostScript Code Page

You may customize the code page files or change the default code page to meet the needs of your installation.

The code page files contain 2-digit hexadecimal code points and their corresponding PostScript character identifier. The code point must occupy columns 1 and 2 in the code page file. The PostScript character identifier must start in column 4 and must not exceed 128 characters. These character names must match the PostScript Font character names in the PostScript AFM files. If column 4 is blank, an error message is issued when the code page is used. If the PostScript character identifier exceeds 128 characters in length, it will be truncated to 128 characters. No DCF error will result, but an error could result at print time. An asterisk in the first column indicates a comment line. Everything on a comment line is ignored.

Some PostScript printers require that the character identifiers used to create composite characters be assigned code points. For example, the lowercase *i acute* (*i*) character is constructed using the dotless *i* and acute characters. On some printers, the *i acute* character cannot be used unless the dotless *i* and acute characters are also assigned code points. These character identifiers are defined in the code page files shipped with DCF. You may assign these characters different code points, but it is recommended that you do not remove them from the file.

Note: Code point 41 is reserved for the 'required blank' character, and any character identifier assigned to code point 41 will be ignored unless the .DC [Define Character] control word is used to change the default 'required blank' character.

The following example shows excerpts from PostScript code page PSCP0395. The entry preceded by '*' represents a character that is commented out because a corresponding character does not exist in the Adobe font character set.

Note: Code points '38' through '3F' in this example show characters needed by some printers to construct composite characters.

```
38 acute
39 cedilla
3A circumflex
3B grave
3C tilde
3D dieresis
3E dotlessi
3F ring
40 space
42 acircumflex
43 adieresis
44 agrave
45 aacute
46 atilde
47 aring
48 ccedilla
49 ntilde

:
F0 zero
F1 one
F2 two
F3 three
F4 four
F5 five
F6 six
F7 seven
F8 eight
F9 nine
**FA seveneighths
FB Ucircumflex
FC Udiereis
FD Ugrave
FE Uacute
```

Figure 22. Example of a PostScript Code Page

Note: Do not assign any code points below hexadecimal '20'. These characters are reserved for ASCII control codes.

The PostScript DCFINDEX File

The PostScript DCFINDEX file associates the AFM filename with the typeface family name and other font characteristics. This file allows the user to descriptively define fonts with the .DF control word when formatting for PostScript devices.

You cannot generate a PostScript DCFINDEX file by running the Font Library Index Program, as you do with IBM fonts. You maintain the DCFINDEX file by using a text editor to add, delete, or customize font entries.

You can add comments to the DCFINDEX file by putting an asterisk (*) in the first column on any line. A DCFINDEX file without comments processes faster than a file with comments.

The structure of the DCFINDEX file is shown in Figure 23.

Typeface Family	Width	Weight	Italic	Underscore	Outline	Overstrike	AFM Filename
* *							
Times Roman							PSFTR
Times Roman	7						PSFTRB
Times Roman		x					PSFTRI
Times Roman	7	x					PSFTRBI
Courier							PSFCR
Courier			x				PSFCRO
Courier	7						PSFCRB
Courier	7	x					PSFCRBO
Helvetica							PSFHV
Helvetica			x				PSFHVO
Helvetica	7						PSFHVB
Helvetica	7	x					PSFHVB0
Helvetica	3						PSFHVL
Helvetica	3	x					PSFHVLO
Helvetica	8						PSFHVBK
Helvetica	8	x					PSFHVBK0
* *							
Symbol							PSFSYM
ITC Avant Garde Gothic							PSFAGB
ITC Avant Garde Gothic			x				PSFAGBO

Figure 23. Example of Font Entries in the DCFINDEX File

The information in DCFINDEX is column sensitive. The values for each column are described below:

Columns 1–32

Contain the typeface family name used on the .DF TYPE specification. This string can include blanks, but it cannot include more than one blank between words.

Column 34

Contains a value 1–9 or a blank to specify the width of the font. The range of values corresponds to the width values on the .DF [Define Font] control word as follows:

- 1 = ultracondensed
- 2 = extracondensed
- 3 = condensed
- 4 = semicondensed
- 5 = normal
- 6 = semiexpanded
- 7 = expanded
- 8 = extraexpanded
- 9 = ultraexpanded

If column 34 is blank, the default of 5 (normal) is used and no error message is issued. If column 34 does not contain a blank or a value between 1 and 9, a warning message is issued and the default of 5 (normal) is used.

Column 36

Contains a value 1–9 or a blank to specify the weight of the font. The range of values corresponds to the weight values on the .DF control word as follows:

- 1 = ultralight
- 2 = extralight
- 3 = light
- 4 = semilight
- 5 = medium
- 6 = semibold
- 7 = bold
- 8 = extrabold
- 9 = ultrabold

If column 36 is blank, the default of 5 (medium) is used and no error message is issued. If column 36 does not contain a blank or a value between 1 and 9, a warning message is issued and the default of 5 (medium) is used.

Column 38

Indicates whether the font has the italic attribute. If this column contains a nonblank character, the font is assumed to be italic, and you must specify the italic attribute on the .DF control word.

Column 40

Indicates whether the font has the underscore attribute. If this column contains a nonblank character, the font is assumed to be underscored, and you must specify the underscore attribute on the .DF control word.¹⁴

Column 42

Indicates whether the font has the outline attribute. If this column contains a nonblank character, the font is assumed to be outlined, and you must specify the outline attribute on the .DF control word.¹⁴

Column 44

Indicates whether the font has the overstrike attribute. If this column contains a nonblank character, the font is assumed to be overstruck, and you must specify the overstrike attribute on the .DF control word.¹⁴

¹⁴ None of the 43 AFM files provided with DCF has the underscore, outline, or overstrike attributes.

Columns 46–53

Contain the corresponding AFM file name, which can be up to 8 characters long. If column 46 is blank, a warning message is issued to tell you that the AFM file name is missing, and the entry is ignored.

The dataset names for the DCFINDEX file coincide with the font library naming conventions in the various operating environments:

CMS	DCFINDEX FONTPS <i>fm</i>
MVS	SCRIPT.R40.FONTPS(DCFINDEX)
VSE	The sublibrary is FONTPS. The phase name is DCFINDEX.

Adding DCFINDEX File Entries

To make a new AFM file accessible to the SCRIPT/VS formatter, you must add an entry to the DCFINDEX file containing the typeface family name, the AFM file name, and any applicable font attributes. When you add the entry, group the typeface family names. If you do not group identical typeface families, those entries of the same typeface family found later in the file will result in an error if they are used.

Suppose you upload the AFM file for a Times Roman underscored font with an AFM file name of PSFTRU. You could add an entry to the DCFINDEX file as shown below. Note that the underscore attribute is specified for the Times Roman font. The underscore attribute is not to be confused with the US parameter on the .DF control word.

Times Roman		PSFTR	
Times Roman	x	PSFTRU	<-- New entry
Courier		PSFCR	

In a document, the user would define this font in 20 point type as follows:

```
.df tru type('times roman' underscore 20)
```

Alternately, you could add the Times Roman underscored font to the DCFINDEX file as shown below. Note that *underscore* is now part of the typeface family name and that the underscore attribute is not specified.

Times Roman		PSFTR	
Times Roman underscore		PSFTRU	<-- New Entry
Courier		PSFCR	

In a document, the user would define this font in 20 point type as follows:

```
.df tru type('times roman underscore' 20)
```

Note that the word *underscore* must be inside the single quotation marks because it is part of the typeface name. If it is not inside of them, it will be recognized as an attribute and will not match the DCFINDEX entry.

Performance Considerations

- Group typeface family names.
- Put the more frequently used typeface families at the top of the DCFINDEX file.
- To specify a default value for the width and the weight fields, use a blank instead of the numeric value of 5.

Customizing DCFINDEX File Entries

DCF has a limited number of font-weight identifiers, font-width identifiers, and font-attribute identifiers. Not all PostScript font characteristics correspond exactly to these identifiers; therefore, one-to-one mapping is not always possible.

For example, the range of values allowed in the DCFINDEX file makes it possible to specify an italic attribute for a Helvetica font entry. In the DCFINDEX file shipped with DCF, this attribute actually maps to the Helvetica *oblique* AFM file.

You may want to customize the DCFINDEX file by changing font mappings. For example, the following AFM files are supplied with DCF:

- PSFBKL (ITC Bookman light)
- PSFBKD (ITC Bookman demibold)

The entries for these fonts in the DCFINDEX file shipped with DCF look like this:

ITC Bookman	3	PSFBKL
ITC Bookman	6	PSFBKD

Note that the light weight attribute (3 = light) maps to the Bookman light AFM file (PSFBKL) and the semibold weight attribute (6 = semibold) maps to the Bookman demibold AFM file (PSFBKD). Using the DCFINDEX file shipped with DCF, the user would define a light Bookman font like this:

```
.df bk type('ITC Bookman' light)
```

and a demibold Bookman font like this:

```
.df bk type('ITC Bookman' semibold)
```

This mapping requires that users know the characteristics of light, demibold, and semibold fonts. Most users know the difference between a medium and a bold font; however, they may not understand more complex typographic concepts.

To simplify these font definitions, you could customize the font entries in the DCFINDEX so that users request the default weight of medium instead of specifically asking for a light font. You could also customize the font entries so that users request a bold weight and get a semibold font. Customized entries in the DCFINDEX file are shown in the example below:

ITC Bookman		PSFBKL
ITC Bookman	7	PSFBKD

Note that default weight attribute (blank = medium) now maps to the Bookman light AFM file (PSFBKL) and the bold weight attribute (7 = bold) now maps to the Bookman demibold AFM file (PSFBKD). The user would define the light Bookman font like this:

```
.df bk type('ITC Bookman')
```

and the demibold Bookman font like this:

```
.df bk type('ITC Bookman' bold)
```

If a new medium-weight ITC Bookman font becomes available and you want to make that AFM file the default, you can easily change the entries in the DCFINDEX file.

Configuring Your Printer for 8-Bit ASCII

The ASCII PostScript data stream produced by DCF uses the high-order bit to produce characters above hexadecimal '7F' in your output file. Your printer must be set up to handle the 8-bit data stream, or you will get printer errors. The IBM 4216 Personal Pageprinter needs no modification to handle 8-bit ASCII. Check your printer documentation to see if your printer can be configured to accept 8-bit ASCII.

Following is an example of the PostScript programming language **setsccbatch** command that can be used with PostScript interpreters to correctly configure the printer's parity setting for 8-bit data:

```
serverdict begin 0 exitserver
statusdict begin
25 9600 7 setsccbatch
♦
```

The "25" and "9600" in the **setsccbatch** command indicate you are setting the parity for the 25-pin channel with 9600 baud. The "7" indicates no parity with Data Terminal Ready (DTR) flow control. The ♦ represents a **Control-d** character that is a hexadecimal '04', which indicates End-of-File. This sample PostScript program allows large 8-bit ASCII files to print.

If your PostScript interpreter does not have DTR flow control, the parity can be set to 3 instead of 7, but this will limit the size of your PostScript files. Check your printer documentation for more information about the serial communication capability of your hardware.

PostScript Output Structuring Conventions

When SCRIPT/VS produces output for a PostScript device, the output follows the applicable Version 2, Adobe Document Structuring Conventions. These structuring conventions can be used by document handlers and font servers to manage the document resources, such as paper size and fonts.

These structuring conventions are actually PostScript comments. SCRIPT/VS uses the following subset of the Adobe Document Structuring Conventions:

%!PS-Adobe-2.0	%%EndProlog
%%BeginDocument:	%%EndSetup
%%BeginSetup	%%IncludeFont:
%%BoundingBox:	%%Page:
%%CreationDate:	%%PageBoundingBox:
%%Creator:	%%PageFonts:
%%DocumentFonts:	%%Pages:
%%DocumentNeededFonts:	%%PaperSize:
%%DocumentPaperSizes:	%%Title:
%%EndComments	%%Trailer
%%EndDocument	

PostScript Command Abbreviations

In order to increase efficiency, SCRIPT/VS defines and uses the following abbreviations for the PostScript commands listed:

AR	array	PP	pop
AS	ashow	PT	put
AW	awidthshow	RL	rlineto
CF	currentfont	FM	rmoveto
CL	closepath	RO	rotate
CM	cleartomark	RS	restore
CP	copy	SC	scale
DF	def	SF	setfont
DP	dup	SG	setgray
FF	findfont	SH	show
FL	fill	SP	showpage
GR	grestore	ST	stroke
GS	gsave	SV	save
LW	setlinewidth	TR	translate
MK	mark	WS	widthshow
MV	moveto	XF	scalefont
NP	newpath		

Other PostScript Considerations

When formatting for and printing on PostScript devices, there are some things you should consider:

- All PostScript printers reserve an unprintable area around the edges of the physical sheet of paper. This unprintable area is used by the tracking mechanisms that move the physical sheet of paper through the paper path. Neither SCRIPT/VS errors nor printer errors result if characters are placed in this unprintable area, but output results may be undesirable. You can modify the DCF logical device table to set page lengths and page widths so that your end users will be unable to place characters in these unprintable areas.
- Even though 43 AFM files are shipped with DCF, your printer may be able to handle only a subset. If you format your document with a font that is not resident on your printer or was not downloaded to your printer, the printer will print the document in Courier. Also, because the document was formatted with different font metrics, the placement of the text on the output page will be incorrect.

- Ensure that any modifications to the code page files, AFM files, and PostScript DCFINDEX file follow DCF and Adobe conventions. If they do not, you may receive unpredictable results.
- SCRIPT/VS imposes no limit on the number of fonts per page. However, your printer may run out of storage if too many fonts are used.
- For IBM printers, all positioning measurements are relative to the top-left corner of the page. For PostScript printers, the positioning measurements are relative to the bottom-left corner of the page. Therefore, if the logical page length is smaller than the physical page length, the logical page will be positioned at the top of an IBM printer physical page and at the bottom of a PostScript physical page. Likewise, if the logical page length is larger than the physical page length, the logical page will extend beyond the bottom of an IBM printer physical page and extend beyond the top of a PostScript physical page.

Appendix F. DCF Programming Interface for CICS

Appendix F. DCF Programming Interface for CICS	291
Interface Description	292
General Sequence of Events	292
Detailed Description	292
Calling Protocol	292
Host Initialization	293
SCRIPT/VS Options Check	295
Host Command Check	295
SCRIPT/VS Formatting	296
SCRIPT/VS Cleanup	302
Host Cleanup	303
Other Considerations	303
Terminal Output Considerations	303
Device Type Determination	303
CICS Services Used by SCRIPT/VS	304
CICS Generation	304
Generalized Markup Language Macros	304
Font Library Support	305
Page Segment Library Support	305
Additional Interface Control Block Information	305
Interface Control Block	306
Command Options Block	309

Appendix F. DCF Programming Interface for CICS

The Document Composition Facility (DCF) provides an Application Programming Interface (API) to the SCRIPT/VS formatter for Customer Information Control System (CICS) applications such as the Advanced Text Management System-III (ATMS), a family of IBM licensed programs: ATMS-III/OS/VS, 5740-XYL, and ATMS-III/VSE, 5746-XXU. Although the interface was designed specifically to support ATMS, it can be used by any CICS/MVS or CICS/VSE application to call the SCRIPT/VS formatter as an independent subroutine.

This appendix is intended to help the text programmer call the SCRIPT/VS formatter as an independent subroutine through CICS/MVS or CICS/VSE application. This appendix documents this general-use programming interface and its associated guidance information provided by DCF.

General-use programming interfaces allow the text programmer to write programs that request or receive the services of DCF.

The SCRIPT/VS formatter may be called as an independent subroutine in the CICS environments by using the interface standards described in this appendix. Before calling the SCRIPT/VS formatter, the caller must build an Interface Control Block (see "Interface Control Block" on page 306) that contains:

- A list of pointers to various service routines and data areas
- Values that represent caller-specifiable parameters to SCRIPT/VS
- Save areas.

The interface requires all CICS applications to provide the following service routines:

- Input services to provide input data to the SCRIPT/VS formatter
- Output services to accept formatted output data from SCRIPT/VS.

The following services are optional:

- File services to support the FILE command option
- Message services to permit SCRIPT/VS to send all messages to the CICS application
- Terminal input/output services to permit SCRIPT/VS to read from and write directly to a user terminal
- Host Command services to permit SCRIPT/VS to communicate (with the .SY control word) to the CICS application
- Host Command Check services to provide a means for the application to complete the checking of the SCRIPT/VS command options

- If page printers, such as the IBM 3820 Page Printer, are supported, the interface requires that the application provide service routines for:
 - Font services to permit SCRIPT/VS to read font library members
 - Page segment services to permit SCRIPT/VS to read page segments.
- If PostScript output is supported, the interface requires that the application provide a font services routine to permit SCRIPT/VS to read the DCFINDEX member and the Adobe Font Metrics (AFM) files. For more information see Appendix E, "PostScript Information" on page 277.

Interface Description

In the description that follows, the CICS application is referred to as the host.

General Sequence of Events

Following is the general sequence of events:

1. The host performs initialization of the Interface Control Block (ICB) and then invokes SCRIPT/VS as a subroutine via a CICS LINK to DSMATS30.
2. SCRIPT/VS performs the initial options check and then uses, if available, the host command-check routine (DSMEHOST) to complete validation of the command options.
3. SCRIPT/VS formats the document:
 - a. Uses input services (DSMEIOP) to get input
 - b. Completes an entire page
 - c. Uses output services (DSMEFOP) to send the formatted page (one record at a time) to the host
 - d. Uses other optional routines as required and available.

The above steps are repeated until the document is formatted.

4. SCRIPT/VS performs cleanup and returns to the host via a CICS RETURN.
5. The host is required to:
 - a. Handle any delayed messages not formatted by SCRIPT/VS
 - b. Purge any remaining temporary storage queues (DSMQNAME)
 - c. Free the ICB area.

Detailed Description

Calling Protocol

The host invokes SCRIPT/VS via a CICS link to module DSMATS30 whose entry point is DSMESBEG. At the time of the link, register 1 must contain the address of the Interface Control Block (ICB).

Other than the initial link to SCRIPT/VS and the final CICS return from SCRIPT/VS to the host, SCRIPT/VS uses a BALR interface to the host:

```

L      R14,xxxxxxx      GET HOST ADDRESS
STM    R14,R11,DSMSAV2  SAVE MY REGS
LM     R15,R11,DSMSAV1+4 RESTORE HOST REGISTERS
BALR   R14,R14          CALL THE HOST FOR SERVICE
LM     R14,R11,DSMSAV2  RESTORE MY REGS

```

The host is expected to:

1. Perform the requested service.

2. Set DSMCODE to one of the following:

DSMCOK (0) No errors. Also set DSMREAS = DSMROK.

DSMWARN (4) Warning. Set DSMREAS to one of the following:

DSREOF (End Of File)

DSRDNF (Document Not Found)

DSRFLUD (Font Library Index Undefined)

DSRFLNF (Font Library Member Not Found)

Depending upon the operation being performed, SCRIPT/VS may upgrade the error code to 8 or 12.

DSMCTERR (16) Terminate operation. Set DSMREAS to any applicable reason code, including the ones normally used for a warning type of error.

3. Return to SCRIPT/VS via register R14.

When SCRIPT/VS has completed all of its operations, it returns control to the host via a CICS RETURN to the host instruction following the CICS LINK to SCRIPT/VS. Return and reason codes are returned in DSMCODE and DSMREAS. DSMCODE contains the highest error that SCRIPT/VS encountered.

Host Initialization

The host is required to perform the following before calling SCRIPT/VS:

1. Get storage (cleared to binary zeroes) for the ICB.

2. Put the address of the ICB into register 1.

3. Create a character string containing the input document name followed by the SCRIPT command options. That string has the following form:

(docname) (options)

(docname) docname is the name of the primary input file. The parentheses are required if the docname contains special characters (including blanks) or if it has more than 8 characters.

(options) options may be any of the SCRIPT command options supported in the CICS/ATMS environment

4. Put the address of the document name and options string into DSMCDP and the length of it into DSMECDPL.

Note: Setting DSMCDP and DSMECDPL to 0 is equivalent to pointing to a command character string consisting of a single question mark (?) that requests SCRIPT/VS to simply list the supported command options in the CICS/ATMS environment. The list is generated as a series of messages.

5. The interface is designed for a host that knows where the output is to be directed. Therefore, the host is expected to ensure that the command options string is ended by the correct device type. For example:
 - For terminal devices: DEV(TERM) or DEV(2741)
(See "Terminal Output Considerations" on page 303 for additional information.)
 - For CTF output to be used as input to STAIRS: DEV(STAIRS). Storage and Information Retrieval System (STAIRS) is an IBM licensed program.
6. Put addresses of the required service routines into the ICB:

DSMEIOP: An input routine is always required.

DSMEFOP: A formatted output routine is always required.
7. Put addresses of any optional service routines into the ICB:

DSMEOUT: A file output routine is required if the host wants to support the FILE command option.

DSMEMOM: A message-handling routine is required if the host wants to handle nondelayed messages.

DSMETIO: A terminal input/output routine is required if the host wants to support direct reads and writes to terminals.

DSMEHCM: A host command routine is required if the host system (.SY) control word is to be supported.

DSMEHOST: A host command-check routine is required if the host wants to perform command-option checking after SCRIPT/VS.

DSMEFLS: A font library routine is required if the host wants to support a page printer or a PostScript device.

DSMEPLS: A page segment library routine is required if reading of page segments for page printers is to be supported.
8. Set required values into the Data Area of the ICB:

DSMHSTID: Temporary storage queue qualifier. This is a host-supplied identifier that SCRIPT/VS uses in creating CICS auxiliary storage queue names. A unique identifier should be supplied by the host for each concurrent SCRIPT operation.

DSMSMAX: Maximum secondary storage acquisitions. This is a host-supplied count of the maximum number of additional secondary storage areas that SCRIPT/VS should acquire over and above its initial storage acquisitions. DSMSMAX provides a means for the host to limit SCRIPT/VS usage of storage.

For practical purposes, DSMSMAX should be at least 50 (the same as the ATMS-III default). If DSMSMAX is zero, SCRIPT/VS obtains an unlimited number of secondary storage areas. The secondary storage areas are currently 8,216 bytes.

If a CICS short-on-storage condition is encountered, SCRIPT/VS terminates processing with DSMREAS set to DSMRISS (Insufficient Storage). If SCRIPT/VS requires more

secondary storage areas than permitted by DSMSMAX, it terminates its processing with DSMREAS set to DSMRSAE (Storage Allocation Exceeded).

Note: SCRIPT/VS uses DSMSCUR to keep count of the number of secondary CICS storage areas that it acquires. This count may be used by the application for accounting and statistical purposes.

9. Invoke SCRIPT/VS as a subroutine with a CICS LINK to DSMATS30.

SCRIPT/VS Options Check

The SCRIPT/VS command options check consists of the following:

- Parsing and converting the command string to uppercase (except for data contained in quotes).
- Checking syntax of the command options and most of the suboptions. The host is responsible for syntax checking of some of the suboptions. See "Host Command Check" for more-detailed information.
- Performing certain device-initialization tasks.

In performing the above, SCRIPT/VS may use certain host service routines, for example:

- Message Handler to process messages
- Input Services to get an OPTIONS document
- Font Library Services for page printer or PostScript device initialization.

Host Command Check

After SCRIPT/VS has checked the options string and has found no severe errors, control passes to the host-command check routine if it is available.

This routine can be used to perform syntax checking of the command options in the Command Options Block (COB) that is pointed to by the address in DSMECOBP. (See "Command Options Block" on page 309.) Because SCRIPT/VS has already processed the options, no modification should be made to the COB.

The host can check the command options to:

1. Determine if the user, with the SCRIPT options, has requested a nonsupported function. For example, if the FILE command option is not going to be supported, the host should check to see if COBSW1 (COBFILE) has been set. If it has, then the command-options string contains a FILE option, and the host should take an appropriate error action.
2. Access and check the syntax of certain names that may be specified as suboptions:

- COBFNAME:** Contains the pointer to the output file name specified by FILE (name).
- COBFNUM:** Contains the size of the file name pointed to by COBFNAME.
- COBFLNAM:** Contains the pointer to the font library name specified by FONTLIB (name). This is a single token of up to 8 alphanumeric characters, padded with blanks.

COBPSNAM: Contains the pointer to the page segment library name specified by SEGLIB (name). This is a single token of up to 8 alphanumeric characters, padded with blanks.

The host is responsible for the use and validity of these names to support SCRIPT/VS

3. Complete the checking of certain SCRIPT command options:

COBLNAME: Contains the pointer to a list of up to 8 tokens specified by LIB (lib1 lib2 ... lib8). Each token consists of 8 alphanumeric characters, padded with blanks.

COBLNUM: Contains the number of tokens in the COBLNAME list.

COBSNAME: Contains the pointer to a list of up to 8 tokens specified by SEARCH (src1 src2 ... src8). Each token consists of 8 alphanumeric characters, padded with blanks.

COBSNUM: Contains the number of tokens in the COBSNAME list.

The host is expected to support the LIB and the SEARCH command options in a manner consistent with ATMS:

LIB The LIB option should be used to specify a hierarchical search order of library data sets that contain symbols and macros (including Generalized Markup Language macros). Library lookup for symbols and macros should be performed for input requests when DSMEFLG2 = DSMSYMAC.

If the host supports only a single symbol and macro library, then the LIB command option is not required.

SEARCH The SEARCH option should be used to specify a hierarchical search order of source input data sets for imbeds and appends. Search lookup should be performed for input requests when DSMEFLG2 = DSMIMAP.

If the host supports only a single library for source documents, then the SEARCH command option is not required.

SCRIPT/VS Formatting

SCRIPT/VS formats the document by obtaining input from the host input services until it has completely formatted one page of output. Then the host formatted output services or file output services is used to pass the output (one record at a time) to the host. As required, and available, SCRIPT/VS uses the other host-supplied routines such as message handling and terminal input/output.

This section provides additional details on what is expected from the following host service routines:

- Input
- Formatted Output
- File Output
- Message Handling
- Terminal I/O

- Host Command
- Font Library
- Page Segment Library.

Input Services: The address of this routine must be in the ICB at DSMEIOP. The purpose of this routine is to provide input services for the SCRIPT/VS formatter. It may be asked to open, read, or close processing on various input sources.

When SCRIPT/VS requires a line of input, it sets DSMEREQ to one of the following:

DSMOPEN	Locate and open the input source identified by DSMEFLG2.
DSMNEXT	Get the next sequential input record.
DSMSPEC	Get the specified input record.
DSMCLOS	Close the input file.

With each OPEN request the ICB contains the following:

DSMNAME:	Contains the name of the requested source input file.
DSMLNAM:	Contains the length of the name in DSMNAME.
DSMEFLG2:	Identifies the type of input source that is being requested: <ul style="list-style-type: none"> • DSMPINP—primary-input source. The name is what is specified in the command-options string. • DSMIMAP—imbedded/appended source. The name is specified on an .IM [Imbed] or .AP [Append] control word. The name may be qualified by the use of a .DD control word. • DSMSYMAC—symbol or macro. Symbol and macro names are folded to uppercase. • DSMOPTS — OPTIONS document. This name is specified with OPTIONS (name) in the command-options string. • DSMPROF — PROFILE document. This is either the default name or the name specified with PROFILE (name) in the command-options string.

For all input requests, DSMEFCA contains a pointer to a 256-byte File Control Area (FCA). If DSMEREQ = DSMOPEN or DSMNEXT, the host is expected to put sufficient information (in any format) into the FCA to permit it to satisfy any subsequent GET SPECIFIED input request from SCRIPT/VS. When that request is issued, the host is expected to use the data in the FCA to do the following:

- Access, as necessary, that particular input source.
- Read, as necessary, until the specified input record is located.
- Point to the specified input record.

For all requests other than DSMEREQ = DSMCLOS, the input routine is expected to update the ICB as follows:

- DSMEINP:** Contains the address of the input record. For an OPEN request, this is the address of the first record.
- DSMELEN:** Contains the length of the input record (maximum value is 256).
- DSMESOF:** Contains the offset into the input record of any sequence-numbering field that SCRIPT/VS is to skip over. This value should always be 0.
- DSMESLN:** Contains the length of any sequence-numbering field in the input record that SCRIPT/VS is to skip over. The maximum value is 16. The data in the sequence-number field is used "as is" if the NUMBER option was specified. DSMELEN should contain the length of both the sequence number field and the input data that follows it.

Formatted Output Services: The address of this routine must be in the ICB at DSMEFOP. The purpose of this routine is to accept a record of formatted output from SCRIPT/VS and route it to its final destination. The ICB contains the following:

- DSMOPTR:** Contains the pointer to the output record.
- DSMOLEN:** Contains the length of the output record.
- DSMEREQ:** DSMPUTL indicates that an output record is ready to be printed.
- DSMEFLG2:** DSMPRNT indicates that this is a formatted print record (as opposed to a condensed text record).
- COBSW4:** COBPS indicates that this is a PostScript output record.
- DSMEFLG1:** Used to further qualify the type of output:
- DSMCTFO indicates that the output record is a condensed text format record with a fixed length of 1008 bytes.
 - DSMPAG indicates that the output record is for a page printer with a variable length record of up to 32K bytes.
 - DSMTRC indicates that the output record is for a 3800 Printing Subsystem Model 1 that contains text reference characters (TRCs).
 - DSMNPA indicates that the output record is the first one for a given page.

Note: Although DSMOPRT (Open Processing on Printer File) has been defined as a valid request code for DSMEREQ, it is not used. Instead, SCRIPT/VS expects the host to open the print file during one of the following times:

- As part of its initialization process.
- As part of its command-check process.
- When the first PUT PRINT request occurs (PUT implies OPEN).

See "Terminal Output Considerations" on page 303 for additional information concerning terminal output.

File Output Services: If this optional routine is provided, its address must be put into DSMEOUT. The purpose of this routine is to accept a record of formatted output when the FILE command option has been specified. The ICB contains the following:

DSMOPTR:	Contains the pointer to the output record.
DSMOLEN:	Contains the length of the output record.
DSMEREQ:	DSMPUTF indicates that a record is ready to be written to an output file.
DSMEFLG2:	DSMFILE indicates that this is a file operation.
DSMNAME:	Contains the name of the output file.
DSMLNAM:	Contains the length of the name in DSMNAME.

If this routine is not provided and FILE was specified, then the output is written to a CICS auxiliary storage queue. SCRIPT/VS puts the name of the queue into DSMSOUTQ of the ICB. This queue *will not* be purged by SCRIPT/VS when it has completed its processing. The host is responsible for processing and subsequently purging that queue.

Note: Although DSMOFMO (Open Processing on Output File) has been defined as a valid request code for DSMEREQ, it is not used. Instead, SCRIPT/VS expects the host to open the output file during one of the following times:

- As part of its initialization process.
- As part of its command-check process.
- When the first PUT FILE request occurs (PUT implies OPEN).

Message-Handling Services: If this optional routine is provided, its address must be put into DSMEMOM. The purpose of this routine is to accept messages from SCRIPT/VS. The ICB contains the following:

DSMOPTR:	Contains the pointer to the message.
DSMOLEN:	Contains the length of the message.
DSMEREQ:	DSMPUTM indicates that this is a message operation.

The host is responsible for the final disposition of the message.

If a message-handling routine is not provided by the host, then SCRIPT/VS writes messages to a CICS auxiliary storage queue. The name of that queue is kept in the ICB at DSMMMSGQ. When SCRIPT/VS has completed its processing, the final disposition of the message queue depends on whether SCRIPT/VS was formatting for a page printer device.

- If the output device is a page printer, then SCRIPT/VS reads the message queue, formats the messages, and appends them to the formatted document (delayed message processing).
- Otherwise, SCRIPT/VS does *not* process the messages. Instead, SCRIPT/VS expects the host to process the message queue as part of its cleanup processing.

Terminal Input/Output Services: If this optional routine is provided, its address must be put into DSMETIO. The purpose of this routine is to provide a read/write terminal capability to support certain control words, such as .RD [Read Terminal] and .TY [Type on Terminal]. The request code (DSMEREQ) in the ICB is set to either DSMGETT (Get Line from Terminal) or DSMPUTT (Put Line to a Terminal).

For READ TERMINAL requests, the terminal I/O routine is expected to update the ICB as follows:

DSMEINP: Contains the address of the terminal input line.
DSMELEN: Contains the length of the terminal input line (maximum value is 256).

The host is responsible for transforming the received terminal-input record into a data record that represents the end-user keyed input.

For WRITE TERMINAL requests, no update of the ICB is required. As with all service requests, the host must set DSMCODE and DSMREAS as appropriate.

DSMOPTR contains the pointer to the terminal output data, and DSMOLEN contains the length of the data. The host is responsible for putting that data into the correct terminal-output-record format and for handling all the associated transmission protocols.

If this routine is not available, then SCRIPT/VS simulates a terminal as follows:

Read Terminal (.RD): SCRIPT/VS simulates a .NL [Null Line] control word.

Read Variable (.RV): SCRIPT/VS simulates a .NL [Null Line] control word.

Terminal Input(.TE): SCRIPT/VS simulates an .EF [End of File] control word.

Type on Terminal (.TY): SCRIPT/VS issues it as a message.

Host-Command Services: If this optional routine is provided, its address must be put into DSMEHCM. The purpose of this routine is to provide a means to communicate to the host with the Host System .SY [System Command] control word. The ICB contains the following:

DSMOPTR: Contains the address of the command specified by the .SY [System Command] control word.
DSMOLEN: Contains the length of the command.
DSMEREQ: DSMHCMD indicates that this is a host-system command.

The host is expected to process the command.

Font Library Services: If this optional routine is provided, its address must be put into DSMEFLS. The purpose of this routine is to provide SCRIPT/VS with a means for reading a font library to support a page printer or a PostScript device.

The operation of this routine is similar to that of the input services routine. The difference is that only font members are requested.

With each request the ICB contains the following:

DSMEREQ:	The request code is one of the following:
DSMOPEN	Locate and open a font member.
DSMNEXT	Get the next sequential font record.
DSMCLOS	Close the font member.
DSMNAME:	Contains the name of the requested font member.
DSMLNAM:	Contains the length of the name in DSMNAME.
DSMEFLG1:	The following flags are set:
DSMPAG	Indicates that this is a request to support a page printer. If DSMPAG is not set, then COBSW4 (COBPS) indicates that this is a request to support a PostScript device.
DSMFLIB	Indicates that this is a Font Library Services request.

For all requests other than DSMEREQ=DSMCLOS, the routine is expected to get the requested font member record and put its address into DSMLINP. The records are variable length (up to 32K bytes) with the length contained in the record.

If DSMCODE is not set to either DSMCOK or DSMCTERR, then SCRIPT/VS expects DSMREAS to be set to one of the following:

DSMREOF	Indicates the end of file.
DSMRIOE	Indicates that an I/O error has occurred.
DSMFLUD	Indicates that the font library is undefined.
DSMFLNF	Indicates that the font library member can't be found.
DSMFLOE	Indicates that the font library member can't be read (opened).

SCRIPT/VS ignores the host setting of DSMCODE and sets the return code to 4, 8, or 12. If DSMREAS is not one of those listed above and DSMCODE is not DSMCOK or DSMCTERR, SCRIPT/VS treats the condition as a programming error and exits with message 400.

If this routine is not provided and the device specified is a page printer or PostScript device, SCRIPT/VS issues an error message and ends processing with a return code of 12.

Page Segment Library Services: If this optional routine is provided, its address must be put into DSMEPLS. The purpose of this routine is to provide SCRIPT/VS with a means for reading page segments to support page printers.

The operation of this routine is the same as for the Font Library Services routine. The difference is that the input source must be a page segment.

With each request the ICB contains the following:

DSMEREQ: The request code is one of the following:

- DSMOPEN** Locate and open a page segment.
- DSMNEXT** Get the next sequential page segment record.
- DSMCLOS** Close the page segment.

DSMNAME: Contains the name of the requested page segment.

DSMLNAM: Contains the length of the name in DSMNAME.

DSMEFLG1: The following flags are set:

- DSMPAG** Indicates that this is a request to support a page printer.
- DSMPLIB** Indicates that this is a page segment request.

For all requests other than DSMEREQ=DSMCLOS, the Page Segment Library routine is expected to get the requested record and put its address into DSMLINP. The records are variable length (up to 32K bytes) with the length contained in the record.

If DSMCODE is not set to either DSMCOK or DSMCTERR, then SCRIPT/VS expects DSMREAS to be set to one of the following:

DSMREOF Indicates the end of file.

DSMRIOE Indicates that an I/O error has occurred.

DSMFLUD Indicates that the page segment library is undefined.

DSMFLNF Indicates that the page segment library member can't be found.

DSMFLOE Indicates that the page segment library member can't be read (opened).

SCRIPT/VS ignores the host setting of DSMCODE and sets the return code to 4, 8, or 12. If DSMREAS is not one of those listed above and DSMCODE is not DSMCOK or DSMCTERR, SCRIPT/VS treats the condition as a programming error and exits with message 400.

If this routine is not provided, then either NOSEGLIB should be specified in the command options or .SI [Segment Include] control words should not be used.

SCRIPT/VS Cleanup

When SCRIPT/VS completes its formatting, it processes the messages in the message queue (DSMMSGQ) for page printers. The messages are formatted and appended to the formatted output. After freeing the storage areas that it has obtained from CICS, SCRIPT/VS returns to the host with CICS RETURN. The DSMFINI flag in DSMEFLG1 indicates that SCRIPT/VS has completed its cleanup processing.

Host Cleanup

The host is required to perform the following:

- Process any delayed messages in the message queue (DSMMSGQ)
- Purge any remaining auxiliary storage queues (DSMMSGQ and DSMSOUTQ)
- Free the ICB area.

The host may force SCRIPT/VS to end its processing by performing the following:

- Setting DSMCODE to an appropriate code
- Setting DSMREAS = DSMREND
- Returning to SCRIPT/VS.

SCRIPT/VS then ends its processing, performs cleanup operations, and returns to the host with a CICS RETURN with the DSMFINI flag set in DSMEFLG1.

Other Considerations

Terminal Output Considerations

This interface does not support a 3270 device type. For terminal output, the only valid device type is DEVICE (2741). Specifying DEVICE (TERM) is the same as specifying DEVICE (2741).

The output is a 2741-like data stream containing certain basic telecommunications access method (BTAM) controls: new lines (X'15'), line feeds (X'25'), idles (X'17'), and backspaces (X'16'). Tab characters (X'05') are converted to blank characters. Furthermore, a single output line may "span" records. The host is expected to check the last byte of the output record. If the last byte is a new line character (X'15') or line feed (X'25'), then the record is the last one for a given output line.

BTAM data management, including the adding of necessary idle characters (X'17') to prevent "printing on the fly," is the responsibility of the host.

For other terminal devices, such as a 3278 Display Station or a 3288 Line Printer, the host is expected to transform the output into an appropriate data stream.

Device Type Determination

If it is necessary for the host to be able to determine the exact physical device type for which the output was produced:

- DSMLDTP contains the pointer to a logical device table.
- At offset 8 in the logical device table there is an 8-byte field containing the name (padded with blanks) of the physical device type for which the output was produced.

Note: For terminal output, it is required that the host append DEV(TERM) or DEV(2741) to the command options string.

A better method for determining the device type is for the host to use the ICB and the COB to determine the type of device for which the output was produced. The best method is for the host to support an end-user interface where the host can determine exactly which DEV(XXXX) character string (along with the appropriate PRINT, FILE or CTF) to append to the command options string.

Under no circumstances should the host permit a data stream produced for one physical device to be directed to an incompatible device. For example, sending output created for a 3820 Page Printer to a 1403 Printer is not going to work; the results are unpredictable.

CICS Services Used by SCRIPT/VS

The SCRIPT/VS formatter uses certain CICS services:

- Storage Control via DFHSC TYPE = GETMAIN and TYPE = FREEMAIN.
- Temporary Storage via DFHTS TYPE = PURGE, TYPE = PUTQ, and TYPE = GETQ.
- Program Control via DFHPC TYPE = RETURN, TYPE = LOAD, and TYPE = DELETE.
- Timer via DFHIC TYPE = GETIME.

Note: Even though SCRIPT/VS uses the macro level to interface to CICS, the CICS application programs that interface to SCRIPT/VS should use the command-level interface to communicate to CICS.

SCRIPT/VS does not issue any CICS ABEND or DUMP requests.

CICS Generation

The following modules should be added as CICS PPT entries:

DSMATS30	SCRIPT/VS formatter
DSMDUVER	Dutch Dictionary
DSMEAVER	English American Dictionary
DSMECVER	English Canadian Dictionary
DSMEUVER	English UK Dictionary
DSMFCVER	French Canadian Dictionary
DSMFNVER	French National Dictionary
DSMGEVER	German Dictionary
DSMITVER	Italian Dictionary
DSMSPVER	Spanish Dictionary

In addition, the CICS auxiliary storage should be specified as a direct access storage device (DASD).

Generalized Markup Language Macros

The Generalized Markup Language macros and profile document must be made accessible to the host. Two ways of accomplishing this are:

1. Copy the individual members out of the system-maintained installation libraries into the host macro library.
2. Use the sequential file created by the installation procedures for the Document Composition Facility in the CICS/ATMS environment. There are header

and trailer records for each member. Those records have the following format:

- Header Record

```
HEADER;DSM32DSMxxxxx          SCRIPT/VS32 DSMGML3 APF
```

where “SCRIPT/VS...” starts in position 55 of the record.

- Trailer record

```
!ATMSEOF!
```

A utility could be written to read the header record, discard the trailer record, and store the member with a name of DSMxxxxx. The process would be repeated until the entire sequential file was read and stored.

Font Library Support

The page printer font libraries must be copied from the system font data sets into CICS data sets available to the host. To conserve space, copy only the first FNG (Font Graphics) record of each member. SCRIPT/VS stops reading once it has read the first FNG of any font member. All members should be copied:

- DCFINDEX member
- Coded fonts
- Code pages
- Font character sets.

Refer to the *Print Services Facility Data Stream Reference* for information on the structure and format of a font library and its members.

DCF provides a DCFINDEX file and AFM files for use when formatting for a PostScript device. See your program directory for the names and libraries of these files. These files must be copied into a separate CICS data set available to the host. See Appendix E, “PostScript Information” on page 277 for additional information.

Page Segment Library Support

A copy of all of the page segments must be made available to the host. If this is unacceptable, then the NOSEGLIB command option should be used.

Additional Interface Control Block Information

The following provides additional information about various Interface Control Block (ICB) fields and flags:

- The DSMOFMO (Open Processing on Output File) request code for DSMEREQ is not used.
- The DSMOPRT (Open Processing on Printer File) request code for DSMEREQ is not used.
- The DSMEOP (End of Parm) request code for DSMEREQ is not used.
- DSMTOUTQ (Terminal Output Queue) is not used.
- DSMSAV1 and DSMSAV2 are used by SCRIPT/VS to save and restore registers 14 through 11. DSMSAV1 is for the host registers at the time of the

initial call to SCRIPT/VS. DSMSAV2 is for the SCRIPT/VS registers at the time of a call to a host service routine.

- DSMSAV3 can be used by the host to save and restore registers.
- DSMEDAY and DSMETIME are used by SCRIPT/VS to store the date and time. The date is obtained from the TCA of CICS. The time is obtained via DFHIC TYPE=TIME.
- DSMWRKP contains the address of the work (WRK) area used by the SCRIPT/VS interface routines.
- DSMSABP contains the address of the Save Area Buffer (SAB) obtained by SCRIPT/VS.
- DSMBEGP contains the address of the initial work area obtained by SCRIPT/VS.
- DSMCICSR is used by SCRIPT/VS to save and restore registers 12 and 13.

Interface Control Block

The Interface Control Block (ICB) is defined by an assembler macro, DSMEICB. The ICB is the primary means of communication between the host application and the SCRIPT/VS formatter.

The ICB macro can be found in the DCF installation library. See your program directories.

```
*****
*          INTERFACE CONTROL BLOCK (DSMEICB)          *
*-----*
*
* COPYRIGHT: LICENSED MATERIALS OF IBM
*          5748-XX9 (C) COPYRIGHT IBM CORPORATION 1980, 1987
*
* $MOD(DSMEICB), COMP(XX9), PROD(DCF) :
* $L3=REL3PKG$, HSR1306, 870530, WGW :
*
*****

DSMEICB DS      0D

*****
*          POINTERS - OPTIONS/SERVICE ROUTINES          *
*****
DSMECDP DS      A          PTR - COMMAND OPTIONS
DSMECDPL DS     A          LENGTH OF COMMAND OPTIONS
DSMEIOP DS      A          PTR - INPUT SERVICE
DSMEFOP DS      A          PTR - FMTD OUTPUT SERVICE
DSMEOUT DS      A          PTR - FILE OUTPUT SERVICE
DSMEMOM DS      A          PTR - MSG HANDLING SERVICE
DSMETIO DS      A          PTR - TERMINAL I/O SERVICE
DSMEHCM DS      A          PTR - HOST COMMAND SERVICE
DSMEHOST DS     A          PTR - COMMAND CHECK SERVICE
DSMEFLS DS      A          PTR - FONT LIBRARY SERVICE
DSMEPLS DS      A          PTR - PAGE SEGMENT LIB. SER.
```



```

*****
*          FLAG BYTES/REQUEST CODES          *
*****
DSMEFLG1 DS      X          FLAG BYTE
DSMXFOCV EQU  X'80'        XFO CMD VALIDATION
DSMCTFO  EQU  X'40'        OUTPUT IS IN CTF FORMAT
DSMNPA   EQU  X'20'        FIRST LINE OF NEW PAGE
DSMFINI  EQU  X'10'        SCRIPT/VS IS FINISHED
DSMFLIB  EQU  X'08'        FONTLIB REQUEST
DSMTRC   EQU  X'04'        OUTPUT CONTAINS TRC'S
DSMPLIB  EQU  X'02'        SEGLIG
DSMPAG   EQU  X'01'        PAGE MODE OUTPUT

*****
* NOTE: DSMEFLG2 MUST MATCH DSBTYPE IN ATMS *
*****
DSMEFLG2 DS      X          FLAG BYTE 2
DSMPINP  EQU  X'80'        PRIMARY INPUT SOURCE
DSMIMAP  EQU  X'40'        IMBEDDED/APPEDED SOURCE
DSMSYMAC EQU  X'20'        SYMBOL OR MACRO
DSMOPTS  EQU  X'10'        OPTIONS DOCUMENT
DSMPROF  EQU  X'08'        PROFILE DOCUMENT
DSMFILE  EQU  X'04'        OUTPUT FILE DOCUMENT
DSMPRNT  EQU  X'02'        FORMATTED OUTPUT FILE
*        EQU  X'01'        RESERVED FOR ATMS

DSMEREQ  DS      X          REQUEST CODE
DSMOPEN  EQU  01          LOC/OPEN SRC (DSMEFLG2)
DSMNEXT  EQU  02          GET NEXT SEQ INPUT
DSMSPEC  EQU  03          GET SPECIFIED UNIT
DSMCLOS  EQU  04          CLOSE PROCESSING ON FILE
DSMPUTL  EQU  05          PUTOUT FORMATTED LINE
DSMPUTF  EQU  06          PUTOUT LINE TO FILE
DSMPUTM  EQU  07          PUTOUT MESSAGE
DSMGETT  EQU  08          GET LINE FROM TERMINAL
DSMPUTT  EQU  09          PUT LINE TO A TERMINAL
DSMHCMO  EQU  10          PROCESS HOST SYSTEM COMMAND
DSMOFMO  EQU  11          OPEN PROCESSING ON OUTPUT FILE
DSMOPRT  EQU  12          OPEN PROCESSING ON PRINTER FILE
DSMEOP   EQU  13          END OF PARMS CODE

*****
*          DATA AREA INFORMATION          *
*****
DSMLDTP  DS      A          LOGICAL DEVICE TABLE POINTER
DSMEFCA  DS      A          PTR TO FILE CONTROL AREA
DSMESOF  DS      H          SEQUENCE CNTL FLD OFFSET
DSMESLN  DS      H          SEQUENCE CNTL FLD LENGTH
DSMELEN  DS      H          LENGTH OF INPUT UNIT (0 - 256)
DSMEINP  DS      A          PTR - INPUT LINE
DSMLINP  DS      A          PTR - INPUT LINE FOR PAGE MODE
DSMOLEN  DS      H          LENGTH OF OUTPUT LINE
DSMOPTR  DS      A          PTR - LINE OF FMTD OUTPUT
DSMHSTID DS      CL4        TEMPORARY STORAGE QUEUE QUALIFIER
DSMLNAM  DS      H          DSMNAME LENGTH
DSMNAME  DS      CL64        SOURCE NAME - DOC/SYM/MAC
DSMSMAX  DS      H          MAXIMUM NUMBER OF SECONDARY BLOCKS
DSMSCUR  DS      H          NUMBER OF SECONDARY BLOCKS ACQUIRED

*****
*          RETURN CODE          *
*****
DSMCODE  DS      X          RETURN CODE (BINARY)
DSMCOK   EQU  00          RETURN OK
DSMCWARN EQU  04          WARNING
DSMCERR  EQU  08          ERROR
DSMCSERR EQU  12          SEVERE ERROR
DSMCTERR EQU  16          TERMINATING ERROR

```

```

*****
*          REASON CODE          *
*****
DSMREAS DS    X          RETURN CODE REASON (BINARY)
DSMR0K  EQU    0          NO REASON CODE
DSMREOF EQU   10          END OF FILE
DSMRFLUD EQU   15          FONT LIBRARY MEMBER UNDEFINED
DSMRI0E EQU   20          I/O ERROR
DSMRVER EQU   25          ICB BUILT INCORRECTLY
DSMRDNF EQU   30          DOC NOT FOUND
DSMRFLNF EQU   35          FONT LIBRARY MEMBER NOT FOUND
DSMRDOE EQU   40          DOC OPEN ERROR
DSMRFLOE EQU   45          FONT LIBRARY OPEN ERROR
DSMREND EQU   50          TERMINATE AT USER REQUEST
DSMRQUE EQU   55          TEMPORARY STORAGE QUEUE ERROR
DSMRSAE EQU   60          STORAGE ALLOCATION EXCEEDED
DSMRISS EQU   70          INSUFFICIENT STORAGE AVAILABLE
DSMRDES EQU   80          TRANSIENT DATA ERROR
DSMRSCR EQU   90          ERRORS IN SCRIPT PROCESSING
DSMRIER EQU   95          INITIALIZATION ERROR
DSMRPMAE EQU  100          MAXIMUM FORMATTED PAGES EXCEEDED
*****

*          QUEUE NAMES          *
*****
DSMQNAME DS    0CL24      TEMP STORAGE QUEUE NAMES
DSMMSGQ  DS     CL8        DELAYED MESSAGE QUEUE
DSMSOUTQ DS     CL8        SEQ OUTPUT QUEUE
DSMTOUTQ DS     CL8        TERMINAL OUTPUT QUEUE
*****

*          RETURNED POINTERS FROM SCRIPT          *
*****
DSMECOBP DS     A          PTR - COMMAND OPTIONS BLOCK
*****

*          REGISTER SAVE AREAS          *
*****
DSMSAV1 DS     14F          SAVEAREA FOR CALLERS REGS
DSMSAV2 DS     14F          SAVEAREA FOR SCRIPT REGS
DSMSAV3 DS     14F          SAVEAREA FOR IPP REGS
*****

*          HOLDER FOR THE TIME & DATE OF THIS RUN          *
*****
DSMEDAY DS     CL4          HOLDER FOR DATE OF THIS RUN
DSMETIME DS    CL4          HOLDER FOR TIME OF THIS RUN
*****

*          POINTERS TO VARIOUS FORMATTER WORK AREAS          *
*****
DSMWRKP DS     F          POINTER TO ESDFH WORK AREA
DSMSABP DS     F          POINTER TO SAVE AREA BLOCKS
DSMBEGP DS     F          POINTER TO ESBEG WORK AREA
DSMCICSR DS    2F          HOLD AREA FOR CICS REGS

```

Command Options Block

The Command Options Block (COB) is defined by an assembler macro, DSMECOB.

The COB macro can be found in the DCF installation library. See your program directories.

```
*****
*          COB - COMMAND OPTION BLOCK                      *
*-----*
*
*   COPYRIGHT: 5748-XX9                                     @BA37579
*   LICENSED MATERIALS - PROPERTY OF IBM                   @BA37579
*   (C) COPYRIGHT IBM CORPORATION 1983, 1990               @BA37579
*
*   CHANGE ACTIVITY:                                       *
*   APAR      DATE      COMMENTS                          *
*   PP33053   84/06/18   MACRO LEFT OFF OF PID ATMS ENV TAPES @BA33053
*   PP33225   84/06/25   PUT .SY IN COB, DEFAULT OFF       @BA33225
*   PP37579   84/11/20   LEVEL SET                          @BA37579
*   PP39862   85/01/25   PUT DDUT IN COB, DEFAULT OFF      @BA39862
*   P112824   87/06/26   MACRO LEFT OFF OF PID ATMS ENV TAPES @BA12824
*
*   $MOD(DSMECOB), COMP(XX9), PROD(DCF) :                 *
*   $L3=REL4PKG$, HSR1401, 900520, WGW :                   @L3C  *
*
*****
          SPACE 1
DSMECOB  DS    0D
          SPACE 1
*   *****
*   *          COB VALUES                                *
*   *****
          SPACE 1
COBEBIND DC   F'0'          DEFAULT BINDING FOR EVEN PAGES (OR 0)
COBOBIND DC   F'0'          DEFAULT BINDING FOR ODD PAGES (OR 0)
COBFNAME DC   A(0)          POINTER TO OUTPUT FILENAME
COBFNUM  DC   F'0'          FILENAME SIZE (TSO & CICS ONLY)
COBLNAME DC   A(0)          LIBRARY NAME LIST ANCHOR
COBLNUM  DC   F'0'          LIB NAME SIZE (TSO & CICS ONLY)
COBVARS  DC   A(0)          POINTER TO SYS VARS (OR 0)
COBPAGES DC   A(0)          POINTER TO PAGE RANGES (OR 0)
COBCHARS DC   A(0)          POINTER TO CHARS LIST
          SPACE 1
COBPNAME DC   CL8'PROFILE'   PROFILE NAME (OR BLANK)          @L3C
COBPNAMEL EQU *-COBPNAME     .Physical size of profile name   @4XA+
          SPACE 1
COBPFNA  DC   A(0)          POINTER TO THE REAL PROFILE FILE
COBPNUM  DC   F'0'          LENGTH OF PROFILE NAME
          SPACE 1
          @L3C
COBSCMST DS    0CL8          SEARCH TYPE FOR CMS              @L3C
COBSNAME DC   A(0)          ADDRESS OF SEARCH LIBS
COBSNUM  DC   F'0'          NUMBER OF SEARCH LIBS
COBTNAME DC   A(0)          TEXT LIBRARY NAME LIST ANCHOR
COBFLNAM DC   A(0)          POINTER TO FONT LIBRARY NAME      @L3C
COBPSNAM DC   A(0)          POINTER TO P.S. LIBRARY NAME      @L3C
          SPACE 1
          @L3C
COBPASS  DC   F'00'         NUMBER OF PASSES SPECIFIED       @310F
          SPACE 1
          @L3C
```

```

*          *****
*          *      COB SWITCHES      *
*          *****
SPACE 1
COBSW1 DC X'00'          COMMAND OPTION SWITCHES 1
COBCTF EQU X'80'          1 = OUTPUT TO DSMUTCTF DESTINATION @L3C
COBCONT EQU X'40'          1 = CONTINUE OPTION
COBFILE EQU X'20'          1 = OUTPUT TO FILE DESTINATION @L3C
COBTERM EQU X'10'          1 = OUTPUT TO TERMINAL DESTINATION @L3C
COBNOWAT EQU X'08' *          1 = NOWAIT OPTION *
COBNUM EQU X'04'          1 = 'NUMBER' THE OUTPUT
COBPRINT EQU X'02'          1 = OUTPUT TO PRINTER DESTINATION @L3C
COBQUIET EQU X'01'          1 = QUIET OPTION
SPACE 1
COBSW2 DC X'00'          COMMAND OPTION SWITCHES 2
COBSVER EQU X'80'          1 = ENABLE .SV
COBSTOP EQU X'40' *          1 = STOP BEFORE EACH PAGE *
COBUCASE EQU X'20'          1 = OUTPUT IN UPPERCASE
COBUNF EQU X'10'          1 = UNFORMATTED OUTPUT
COB2PASS EQU X'08'          1 = TWOPASS OPTION
COBPRMPT EQU X'04' *          1 = PROMPT FOR PAGES *
COBCC EQU X'02'          1 = OUTPUT HAS CARRIAGE CONTROLS@L3C
COBASCII EQU X'01'          1 = PSOUT(ASCII) REQUESTED @32PM
SPACE 1
COBSW3 DC X'00'          COMMAND OPTION SWITCHES 3
COBINDEX EQU X'80'          1 = INDEX OPTION
COBNORC EQU X'40'          1 = .RC DISABLED
COBSPIE EQU X'20'          1 = 'NOSPIE' WAS NOT SPECIFIED
COBPROOF EQU X'10'          1 = STAIRS PROOF OUTPUT
COBNODD EQU X'08'          1 = NODDOUT IN EFFECT. @BA39862
COBBD5W EQU X'04'          1 = BIND OPTION SPECIFIED @L3C
COBNOSEG EQU X'02'          1 = NOSEGLIB IN EFFECT
COBSYOFF EQU X'01'          1 = SY OFF IN EFFECT. @BA33225
SPACE 1
COBSW4 DC X'00'          COMMAND OPTION SWITCHES 4
COBCMDV EQU X'80'          ATMS XFO CMD VALIDATION @L3C
COBPS EQU X'40'          POSTSCRIPT DEVICE OUTPUT @32DV
COBDEST EQU X'20'          DBCS FLAG FOR DEST OPTION(TSO) @40L
COBSEPM EQU X'10'          SEPMASTR OPTION SPECIFIED @4SM
SPACE 1
COBTRACE DC X'00'          MESSAGE OPTION FLAGS (OR ZERO)
COBMSTRC EQU X'80'          MESSAGE TRACE - TRACE IMBED STACK
COBMSID EQU X'40'          MESSAGE ID - INCLUDE ID WITH MSG
COBMDLY EQU X'20'          DELAY MESSAGE OPTIONS
COBMSOC EQU X'10'          MESSAGE OC - OLD CODE MESSAGE @HEB
COBMSEN EQU X'08'          MESSAGE EN - ENGLISH MESSAGE @HEB
*
COBSW5 DC X'00'          .COMMAND OPTION SWITCHES 5 @4XA+
COBOOPN EQU B'10000000'          ..1 = OUTPUT FILE IS OPEN @4XA+
XCBOOPN EQU B'01111111'          ...TURN IT OFF @4XA+
* EQU B'01000000'          ..UNUSED @4XA+
* EQU B'00100000'          ..UNUSED @4XA+
* EQU B'00010000'          ..UNUSED @4XA+
* EQU B'00001000'          ..UNUSED @4XA+
* EQU B'00000100'          ..UNUSED @4XA+
* EQU B'00000010'          ..UNUSED @4XA+
* EQU B'00000001'          ..UNUSED @4XA+
*
DC 2X'00'          .UNUSED (DWORD ALIGNMENT) @4SM@4XA+
COBSEPAR DC A(0)          SEPARATION MASTER ARRAY POINTER @4SM
DC 8X'00'          (RESERVED FOR DWORD ALIGNMENT) @4SM
*
DSMECOBL EQU *-DSMECOB          .PHYSICAL SIZE OF COMMAND OPTION BLOCK +
*-----+

```

Appendix G. Algorithmic Hyphenation Interface in DCF

Appendix G. Algorithmic Hyphenation Interface in DCF	313
Introduction	313
Hyphenation Processing in SCRIPT/VS	313
Hyphenation Algorithms in SCRIPT/VS	314
Algorithmic Hyphenation Routine Interface	315
Expanded Interface Capabilities	315
Register Conventions	315
The Parameter List	316
Routine Entry and Exit Protocol	317
Installation Instructions	317
The CMS Environment	317
The TSO Environment	318

Appendix G. Algorithmic Hyphenation Interface in DCF

Introduction

This appendix describes the interface between DCF Release 3.0 (and above) and its algorithmic hyphenation routines. Specific instructions for using new routines are included. This appendix documents this general-use programming interface and its associated guidance information provided by DCF.

General-use programming interfaces allow the text programmer to write programs that request or receive the services of DCF.

Hyphenation Processing in SCRIPT/VS

Hyphenation processing is invoked by SCRIPT/VS under the following conditions:

1. An input word is found that overdraws (that is, does not fit in) the current output line.
2. Concatenation is in effect (for example, .FO ON).
3. The overdrawn word contains no backspaces.
4. Hyphenation is in effect (for example, .HY ON).
5. Hyphenation cannot be avoided by expanding or compressing the word spaces on the line (for example, .HY RANGE .8 1.2).

Hyphenation processing is language-dependent; the current language is that specified by the most recently executed .DL [Dictionary List] control word. You can query the current language by using the &\$SVLG system symbol.

Several independent mechanisms are available within SCRIPT/VS for hyphenation. Hyphenation is attempted with each of these mechanisms (unless specifically disabled) until an acceptable hyphenation point is found. If no acceptable hyphenation point is found, the entire overdrawn word is spilled onto the next output line if there is already at least one word space on the current output line. If no acceptable hyphenation point is found and there are no word spaces on the output line, the current overdraw option is invoked (for example, .FO TRUNC).

The mechanisms available for hyphenation, and the order in which they are used, are:

1. The addenda dictionary (built with .DU ADD) for the current language (given with .DL).
2. The IBM-supplied hyphenation dictionary for the current language.
3. Any user dictionaries (built with the Dictionary Maintenance Programs) identified with .DL.
4. The hyphenation algorithm for the current language.
5. Any required hyphen characters in the overdrawn word.

A hyphenation point returned by one of the mechanisms listed above breaks the overdrawn word only if it meets the criteria set with the .HY control word:

MINWORD	This parameter gives a minimum word length, in characters, for hyphenated words. Words shorter than the MINWORD length are not hyphenated.
MINPT	This parameter gives the minimum number of characters preceding a hyphenation point.
MAXPT	This parameter gives the minimum number of characters following a hyphenation point.
LADDER	This parameter limits the number of consecutively hyphenated output lines.

Hyphenation Algorithms in SCRIPT/VS

Hyphenation support is provided for fifteen languages in SCRIPT/VS. The current language is determined by a keyword of the .DL control word, and it can be queried using the &\$\$VLG system symbol.

DCF provides hyphenation dictionaries for each of the following languages. Each dictionary contains between 10,000 and 15,000 root words.

Keyword Language

DUTH	Dutch
EAM	American English
ECAN	Canadian English
EUK	U.K. English
FCAN	Canadian French
FNAT	French National
GERM	German
ITAL	Italian
SPAN	Spanish

The IBM-supplied dictionaries for the following six languages are empty:

Keyword Language

DAN	Danish
FIN	Finish
ICE	Iceland
NOR	Norwegian
POR	Portuguese
SWE	Swedish

DCF provides a hyphenation algorithm for all the languages.

Language Algorithmic Hyphenator

Danish	DSMDAALG
Dutch	DSMDUALG
English American	DSMEAALG ¹⁵
English Canadian	DSMECALG ¹⁵
English UK	DSMEUALG ¹⁵

¹⁵ The default algorithmic hyphenator is named DSMTRALG, which is the same one that was supplied with Release 3.2. You need to specify these algorithmic hyphenators to get the new ones supplied with Release 4.0.

Finnish	DSMFIALG
Canadian French	DSMFICALG
French National	DSMFNALG
German	DSMGALG
Icelandic	DSMICALG
Italian	DSMITALG
Norwegian	DSMNOALG
Portuguese	DSMPOALG
Spanish	DSMSPALG
Swedish	DSMSWALG

For words with accented characters to be processed correctly by the algorithmic hyphenators, the correct translation must be provided with the DSMACTT macro. See "Accented Characters" on page 171 for an explanation of accented characters.

Algorithmic Hyphenation Routine Interface

The assembler-level programming interface presented to the algorithmic hyphenator routines by SCRIPT/VS is described below. Emphasis is given to register conventions, the hyphenation parameter list, and entry and exit protocols.

Note: This section describes what you need to do if you want to use your own algorithmic hyphenator instead of the IBM-supplied algorithmic hyphenator.

Expanded Interface Capabilities

The algorithmic hyphenation interface used with DCF Release 2 provided the hyphenation routine with the word to be hyphenated and the "last to fit" count. The hyphenation point was accepted back from the interface.

In DCF Release 3.0, the interface was expanded to provide several additional capabilities:

- The number of characters already on the output line is available.
- The number of word spaces already on the output line is available.
- The hyphenation algorithm can change the spelling of the hyphenated word, if necessary.

This expanded interface can be used to program an algorithmic hyphenation routine to replace the IBM-supplied algorithmic hyphenators.

Register Conventions

When SCRIPT/VS calls one of the algorithmic hyphenation routines listed above, the following general registers contain significant information:

R1 Register one contains the address of the field named WORDHYLF in the following hyphenation parameter list:

WORDHYP	DSECT	
WORDHYLG DS	F	LENGTH OF WORD IN WORDHYLG
WORDHYLF DS	X	LAST-TO-FIT CHARACTER IN WORDHYLG
WORDHYBF DS	CL255	WORD TO BE HYPHENATED
WORDHYCH DS	F	NUMBER OF CHARACTERS ALREADY ON LINE
WORDHYWS DS	F	NUMBER OF WORDSPACES ALREADY ON LINE

Each of these fields is described below.

Note: The field WORDHYLG *precedes* the field addressed by Register 1.

R11 Register 11 contains the address of the SCRIPT common control block. This register should not be altered by your routine or by any subroutines it may invoke.

R12 Register 12 is, by convention, the base register used by all SCRIPT modules. Your routine should copy the contents of Register 15 into Register 12 and use it as its base register.

R13 Register 13 contains the address of a preallocated, prechained register save area:

SAVD	DSECT		
SAVFLAG	DS	X	(RESERVED)
SAVMID	DS	CL3	MODULE IDENTIFICATION
SAVBKWD	DS	A	ADDRESS OF PREVIOUS SAVE AREA
SAVFW	DS	A	ADDRESS OF NEXT SAVE AREA
SAVREG	DS	15F	SAVED REGISTERS 14-15, 0-12
SAVWORK	DS	XL184	WORK AREA

Your routine should use this area to save and restore its caller's registers. Several subsequent save areas are preallocated by SCRIPT/VS and chained together using the SAVFW and SAVBKWD pointers. Any subroutines called by your routines may use these additional save areas.

The field SAVWORK is available as a scratchpad for local data storage. This storage is *not*, however, preserved across consecutive calls to the hyphenation routine.

R14 Register 14 contains the address to which your routine should return when hyphenation processing is complete.

R15 Register 15 contains the entry point address of your routine. This value should be copied into Register 12.

The Parameter List

The algorithmic hyphenation parameter list, which is addressed by Register 1 on entry to the routine, contains the following fields:

WORDHYLG On entry, this fullword contains the length, in characters, of the word to be hyphenated. This field is not examined by SCRIPT/VS when the routine returns.

Note: This field *precedes* the address contained in Register 1.

WORDHYLF On entry, this byte contains the number of characters of the word that fit on the output line. The width of the hyphen character has already been considered in computing this value. On return, SCRIPT/VS examines this byte for the hyphenation point. Your routine should place the number of characters to precede the hyphen in this field, or zero if the routine cannot hyphenate the word.

WORDHYBF On entry, this 255-character field contains the word to be hyphenated, left-aligned and padded on the right with blanks. On return, SCRIPT/VS examines this field for any spelling changes caused by hyphenation.

WORDHYCH	On entry, this fullword contains the number of characters already on the output line that are subject to justification. This field is not examined by SCRIPT/VS when the routine returns.
WORDHYWS	On entry, this fullword contains the number of word spaces already on the output line that are subject to justification. This field is not examined by SCRIPT/VS when the routine returns.

Routine Entry and Exit Protocol

The following entry protocol is suggested for algorithmic hyphenation routines:

```
DSMxxALG CSECT
    USING SAVD,R13
    STM  R14,R12,SAVREGS    SAVE CALLER'S REGISTERS
    L    R13,SAVFWO         LOAD ADDRESS OF NEXT SAVE AREA
    USING DSMxxALG,R12
    LR   R12,R15             PREPARE BASE REGISTER
    MVC  SAVMID,=C'xxx'      STORE MODULE ID IN SAVEAREA
    USING WORDHYP,R1
    S    R1,=A(L'WORDHYLG)   POINT AT BEGINNING OF PLIST
```

The following exit protocol is suggested for algorithmic hyphenation routines:

```
    L    R13,SAVBKWD         LOAD ADDRESS OF CALLER'S AREA
    LM   R14,R12,SAVREGS     RESTORE CALLER'S REGISTERS
    BR   R14                 RETURN TO CALLER
```

The value of Register 15 is not significant on return from the hyphenation routine; SCRIPT/VS determines the success or failure of hyphenation by examining the field WORDHYLF in the parameter list. If the value is not zero, hyphenation is considered successful. The value of WORDHYLF is used as the hyphenation point, and WORDHYBF is examined for any spelling changes caused by hyphenation.

Installation Instructions

Procedures for replacing an IBM-supplied algorithmic hyphenator with your own algorithmic hyphenation routine are given below for the two most widely used SCRIPT/VS environments: CMS and TSO.

The CMS Environment

The following procedure is recommended for replacing an IBM-supplied algorithmic hyphenator with your own hyphenation routine in the CMS environment:

1. Compile or assemble DSMxxALG to produce a TEXT file. If your routine calls any other modules, compile or assemble them also.
2. Link and access the SCRIPT/VS Release 4 installation minidisk in READ/WRITE mode as your A-disk.
3. Move the IBM-supplied DSMxxALG to another disk to save it.
4. Copy your DSMxxALG TEXT file onto the A-disk. If your routine calls any other modules, copy them also.

5. Issue the following CMS commands:

```
txtlib del dsmhylib dsmxxalg
txtlib add dsmhylib dsmxxalg
```

If your routine calls any other modules, issue these commands for each of them also.

6. Issue the following CMS command:

```
dsmgnd40
```

This EXEC creates a new SCRIPT MODULE. If you indicate to include your hyphenation routine, it is link-edited into this new module. Otherwise, your hyphenation routine is loaded from DSMHYLIB TXTLIB when needed.

7. If your installation has defined a shared segment for SCRIPT/VS, issue the following CMS command:

```
dsmgns40
```

This EXEC reloads the shared segment. If you indicate to include your hyphenation routine, it is loaded into the shared segment; otherwise, your hyphenation routine is loaded from DSMHYLIB TXTLIB when needed.

8. Copy the new SCRIPT MODULE and DSMHYLIB TXTLIB files onto the public minidisk where SCRIPT/VS normally resides at your installation.

The TSO Environment

The following procedure is recommended for replacing an IBM-supplied algorithmic hyphenator with your own algorithmic hyphenator in the TSO environment:

1. Compile or assemble DSMxxALG to produce an object file suitable as input to the OS/VS linkage editor. If your routine calls any other modules, compile or assemble them also, and append them to the DSMxxALG object file.
2. Find out the name of the DCF Release 4.0 load library that contains the installed IBM-supplied dictionary routines. This library is named SCRIPT.R40.DCFLOAD in the examples below. DCF may have been placed in another load library, such as SYS1.CMDLIB, at your installation.
3. Execute the following job to replace the IBM-supplied algorithmic hyphenator with your own algorithmic hyphenator in the DCF load library.

```
//INSTALL JOB ,SCRIPTVS,MSGLEVEL=(1,1)
//LINKALG EXEC PGM=HEWL,REGION=768K,
//          PARM='AMODE=24,RMODE=24,RENT,XREF,LIST,LET,NCAL'
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//ALGOBJ DD DSN=DSMxxALG.OBJ,DISP=SHR
//DCF DD DSN=SCRIPT.R40.DCFLOAD,DISP=SHR
//SYSLMOD DD DSN=SCRIPT.R40.DCFLOAD,DISP=OLD
//SYSLIN DD *
INCLUDE ALGOBJ
INCLUDE DCF(DSMxxVER)
ENTRY DSMxxVER
NAME DSMxxVER(R)
NAME DSMxxVER(R)
/*
//
```

4. If the language corresponding to your algorithmic hyphenator was link-edited with SCRIPT/VS when DCF was installed, execute the following job to replace the algorithmic hyphenator in the SCRIPT command:

```
//INSTALL JOB ,SCRIPTVS,MSGLEVEL=(1,1)
//LINKALG EXEC PGM=HEWL,REGION=768K,
//          PARM='AMODE=31,RMODE=ANY,RENT,XREF,LIST,LET,NCAL'
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//ALGOBJ DD DSN=DSMxxALG.OBJ,DISP=SHR
//DCF DD DSN=SCRIPT.R40.DCFLOAD,DISP=SHR
//SYSLMOD DD DSN=SCRIPT.R40.DCFLOAD,DISP=OLD
//SYSLIN DD *
INCLUDE ALGOBJ
INCLUDE DCF(DSMTXS40)
      (insert ORDER statements here)
ENTRY DSMESBEG
NAME DSMTXS40(R)
/*
//
```

Note: The current correct linkage editor ORDER statements should be obtained from the DCF installation sample DSMTSDCT in SCRIPT.R40.DCFSAMP.

Appendix H. Related Publication and Products

	Appendix H. Related Publications and Products	323
	Related Products	323
	Publication Library Guide for the Document Composition Facility	324
	DCF Publications	325
	Optional Features	325
	Related Publications	326

Appendix H. Related Publications and Products

This appendix lists the products, publications, and optional features that are related to SCRIPT/VS and GML.

Related Products

The following products are related to DCF. For information about ordering any of these products, contact your local IBM Branch Office.

- **MARKUP:** A PC-based editor that helps you create GML documents. Refer to *MARKUP User's Guide and Tutorial*, which you can order with the MARKUP product, 6476161.
- **Publishing Systems BookMaster:** A host-based application that runs under control of DCF and is designed for high-volume in-house publishing applications. This application contains a superset of the GML Starter Set.
- **Publishing Systems BookManager BUILD and BookManager READ:** Host-based IBM licensed programs that let you create and use online books and documents at your terminal in a VM/CMS system.
- **OfficeVision:** OfficeVision provides an integrated electronic office that delivers a broad range of office functions for business communications with the help of an integrated set of services.
- **Standard Generalized Mark-up Language (SGML) TextWrite OS/2 Edition:** TextWrite is a software program that writers can use to create and modify SGML-compliant documents.
- **TextTagger:** TextTagger is a software program that analyzes electronic documents and inserts tagging to comply with the Department of Defense's Computer-Aided Acquisition and Logistics Support (CALS) initiative.
- **SGML Translator DCF Edition:** A program that parses, validates, and translates SGML documents so they can be formatted and printed by DCF.
- **Print Services Facility (PSF):** A licensed program that combines print data with resources to manage and control data transmitted to IBM page printers.

Publication Library Guide for the Document Composition Facility

The following table lists the Document Composition Facility publications by number as they relate to user tasks. "DCF Publications" on page 325 lists the titles and the order numbers that correspond to the numbers listed in the table.

Number	User Tasks	Typical Audience	Brief Description
(1) (2) (3) (20)	Planning and introducing DCF/DLF	Users, system planners	Provide a general overview of text processing, library facility, and available books.
(3) (4) (5) (12) (16) (20)	Formatting documents (using the GML starter set)	Novice to experienced end users	Provide an introduction to the Generalized Markup Language (GML) starter set and describes the GML starter set tags and SCRIPT/VS messages.
(6)	Creating bar codes with DCF/GML	Experienced end users	Provides information about using GML to create bar codes.
(9) (10) (11) (12) (17) (18) (19) (21)	Formatting documents (using SCRIPT/VS control words)	Knowledgeable to experienced end users	Describe the function and use of all SCRIPT/VS control words, macros, diagnostic aids, and the formatting features and messages.
(14) (15)	Converting RFTDCA for SCRIPT/VS formatting	Novice to experienced RFTDCA users	Describe the function and use of the optional Office Document Feature, including diagnostic aids and messages.
(4) (5) (7) (9) (10) (11) (19) (21)	Modifying GML starter set ¹⁶	Document administrator and text programmer ¹⁷	Contain material about GML starter set tags, SCRIPT/VS control words, and GML starter set modifications.
(4) (5) (7) (8) (9) (10) (11) (16) (19) (21)	Creating GML application processing functions	Document administrator and text programmer ¹⁷	Provide information about designing your own GML and about GML concepts, GML starter set tags, SCRIPT/VS control words, and usage guidelines.
(10) (11) (12) (13) (19) (21) (24) (25) (26) (27) (28)	Installing, modifying, and maintaining DCF	System programmer	Give information on error isolation, program tailoring, and use of SCRIPT/VS.
(22)	Creating mathematical formulas with SMFF	Experienced end users	Describes the function and use of the SCRIPT Mathematical Formula Formatter (SMFF), including .EQ control word and messages.
(23)	Creating memos, transparencies, and schedules with GML applications	Novice to experienced end users	Describes the use of the memo, transparencies, and schedule applications, including messages.

¹⁶ Central Programming Service support and maintenance are provided **only** on the unmodified GML applications shipped with DCF. If you modify any of these GML applications shipped with DCF, it is recommended that you also maintain an **unmodified** copy for diagnostic purposes.

¹⁷ The document administrator is responsible for defining markup conventions and procedures for an organization. The text programmer implements application processing functions (APFs) that provide the processing specified by the document administrator.

DCF Publications

Number	Titles and Order Numbers
--------	--------------------------

- | | |
|------|--|
| (1) | <i>About DCF</i> , G520-6362. |
| (2) | <i>Document Composition Facility and Document Library Facility General Information</i> , GH20-9158. |
| (3) | <i>Document Composition Facility: Introduction to Generalized Markup Language</i> , G544-3192. |
| (4) | <i>Document Composition Facility: Generalized Markup Language Starter Set User's Guide</i> , SH20-9186. |
| (5) | <i>Document Composition Facility: Generalized Markup Language Starter Set Reference</i> , SH20-9187. |
| (6) | <i>Document Composition Facility: Bar Code User's Guide</i> , S544-3115. |
| (7) | <i>Document Composition Facility: Generalized Markup Language Starter Set Implementation Guide</i> , SH35-0050. |
| (9) | <i>Document Composition Facility: SCRIPT/VS User's Guide</i> , S544-3191. |
| (10) | <i>Document Composition Facility: SCRIPT/VS Text Programmer's Guide</i> , SH35-0069. |
| (11) | <i>Document Composition Facility: SCRIPT/VS Language Reference</i> , SH35-0070. |
| (12) | <i>Document Composition Facility: SCRIPT/VS Messages</i> , SH35-0048. |
| (13) | <i>Document Composition Facility: Diagnosis Guide and Reference</i> , LH40-0209. |
| (14) | <i>Document Composition Facility: Office Document Feature User's Guide</i> , G544-3129. |
| (15) | <i>Document Composition Facility: Office Document Feature Reference</i> , S544-3130. |
| (16) | <i>Using the Document Composition Facility</i> , SR21-0515 (Training Course 32291). |
| (17) | <i>Using DCF with the 4250 Printer</i> , SR20-8486 (Training Course 32908). |
| (18) | <i>Using DCF with Page Printers</i> , SR21-1211 (Training Course 32243). |
| (19) | <i>Document Composition Facility—Release 3 (SCRIPT/VS) for Document Administrators and Text Programmers</i> , SR20-7525 (Training Course). |
| (20) | <i>Document Composition Facility (SCRIPT/VS) Student Text</i> , SC20-1894 (Training Course). |
| (21) | <i>Document Composition Facility: TSO Enhancements Update Guide</i> , G544-3345. |
| (22) | <i>Document Composition Facility: SCRIPT Mathematical Formula Formatter User's Guide</i> , S544-3306. |
| (23) | <i>Document Composition Facility: Generalized Markup Language (GML) Applications User's Guide</i> , G544-3305. |
| (24) | <i>Document Composition Facility: MVS Program Directory</i> , G544-3669. |
| (25) | <i>Program Directory for use with DCF and SMFF for VM</i> , G544-3670. |
| (26) | <i>Document Composition Facility: VSE Program Directory</i> , G544-3671. |
| (27) | <i>Document Composition Facility: ODF Program Directory for MVS</i> , G544-3687. |
| (28) | <i>Document Composition Facility: ODF Program Directory for VM</i> , G544-3686. |

The following are also available:

- *Document Composition Facility: GML Starter Set Quick Reference*, SX26-3723.
- *Document Composition Facility: SCRIPT/VS Text Programmer's Quick Reference*, SX26-3719.
- *Document Composition Facility Post-Processor Examples*, S544-3484.
- *Document Composition Facility*, SH35-0086 (binder).

Note: The DCF publications will be available at the end of the second quarter of 1991 as displayable BookManager built BOOKs and as source files on a CD-ROM, SK25-1980.

Optional Features

For information about ordering these optional features, contact your local IBM Branch Office:

- **SCRIPT Mathematical Formula Formatter (SMFF):** An optional feature of DCF that can be ordered separately. SMFF makes it possible to construct and display mathematical and scientific formulas on page printers.
- **Office Document Feature (ODF):** An optional feature of DCF that can be ordered separately. ODF allows documents that have been created with an office system to be printed on DCF-supported printers.

Related Publications

You should use the following publications to evaluate the use of DCF in different operating environments:

- *IBM Virtual Machine/System Product: Introduction*, GC19-6200.

This publication contains an introduction to CMS (Conversational Monitor System), which is one of the interactive systems that SCRIPT/VS operates with.

- *IBM Virtual Machine/System Product: Terminal User's Guide*, GC19-6206.

This publication describes the various terminal types supported by VM/SP for those who plan to use VM/SP in their operations.

- *OS/VS2 TSO Terminal User's Guide*, GC28-0645.

This publication gives detailed user information about OS/VS2 TSO (Time Sharing Option), which is one of the interactive systems that SCRIPT/VS operates with. It describes the TSO EDIT facility and related facilities for text entry, text editing, and data set management.

- *A Guide to IBM's Advanced Function Printing*, G544-3095.

This publication describes the use of a licensed program (PSF, DCF, GML, OGL, GDDM, and PMF) and the use of a subset of a licensed program in conjunction with the IBM Advanced Function Printing (AFP) printers, including the IBM 3800 Printing Subsystem Models 3 and 6 and the IBM 3820 Page Printer.

- *Advanced Function Printing Software: General Information*, G544-3415.

This publication defines Advanced Function Printing (AFP), describes the features and functions of the AFP licensed programs, and shows how the programs work together. It is intended for the people in your organization who will plan for, install, use, and maintain IBM's AFP software products. It also contains a list of the AFP publications.

If you install DLF in the MVS environment, you need a copy of *OS/VS2 Access Method Services*, GC26-3841.

If you install DLF in the VSE environment, you need a copy of *VSE/VSAM Access Method Services: User's Guide and Reference*, SC24-5144.

Glossary

Source Identifiers

This publication includes terms and definitions from:

- The *American National Dictionary for Information Processing Systems*, copyright 1982 by the Computer and Business Equipment Manufacturers Association (CBEMA). Copies may be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *ISO Vocabulary—Information Processing and the ISO Vocabulary—Office Machines*, developed by the International Organization for Standardization, Technical Committee 97, Subcommittee 1. Definitions of published segments of the vocabularies are identified by the symbol (I) after the definition; definitions from draft international standards, draft proposals, and working papers in development by the ISO/TC97/SC1 vocabulary subcommittee are identified by the symbol (T) after the definition, indicating final agreement has not yet been reached among participating members.

References

The following cross-references are used in this glossary:

Deprecated term for

Indicates that the term should not be used (because it is obsolete, misleading, ambiguous, or jargonistic) and refers to the preferred term. For a deprecated term, the commentary contains only this reference; the deprecated term is not defined.

Synonymous with

Appears in the commentary of a preferred term and identifies less desirable or less specific terms that have the same meaning. The commentaries of the less desirable or less specific terms refer back to the preferred term with the *Synonym for* reference words.

Synonym for

Appears in the commentary of a less desirable or less specific term and identifies the preferred term that has the same meaning.

Contrast with

Refers to a term that has an opposite or substantively different meaning.

See Refers to a multiple-word term in which this term appears.

See also

Refers to related terms that have similar (but not synonymous) meanings.

A

Adobe Document Structuring Conventions. A standard subset of conventions that allow PostScript page descriptions to be accepted as input by many programs and resource managers.

Adobe Font Metrics (AFM) files. Files containing PostScript font information that SCRIPT/VS uses to format documents for PostScript devices.

advanced function printing (AFP). The ability of licensed programs to use the all-points-addressable concept to print text and images on a printer.

alignment. The horizontal placement of text in a column or cell.

all-points addressability. The capability to address, reference, and position text, overlays, and images at any defined point on the printable area of the paper. See *page device* and contrast with *line device*.

all-points addressable (APA). In computer printing, pertaining to the ability to address and print or not print each picture element (pel) on a page.

alphameric. Synonym for alphanumeric.

alphanumeric. Pertaining to a character set that contains letters, digits, and usually other characters, such as punctuation marks. (A) Synonymous with alphameric.

alphanumeric character set. A character set that contains both letters and digits and may contain control characters and special characters. (T) Synonymous with alphameric.

alphanumeric character subset. A character subset that contains both letters and digits and may contain control characters, special characters, and the space character. (I) (A) Synonymous with alphameric.

alphanumeric string. A sequence of characters consisting of the letters a through z and the numerals 0 through 9.

ampersand. The & character. When an ampersand begins a character string, SCRIPT/VS assumes the character string is a symbol name. If the symbol

name is defined, SCRIPT/VS replaces the symbol with its value (unless symbol substitution is off).

APF. Application processing function.

application processing function (APF). In GML processing, the processing that is performed when a document element or attribute is recognized. In SCRIPT/VS, an APF is implemented as a sequence of control words, possibly intermixed with text and symbols, in one of three forms: macro definition, value of a symbol, or imbedded file.

application program interface (API). The formally-defined programming language interface which is between an IBM system control program or a licensed program and the user of the program.

ascender. (1) In a font, the distance from the baseline to the top of the character. See *maximum ascender*. (2) The part of a lowercase letter that rises above the body of the letter. Letters with ascenders are b, d, f, h, k, l, and t.

ASCII. American National Standard Code for Information Interchange. The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A) Contrast with *EBCDIC*.

Note: IBM has defined an extension to ASCII code (characters 128–255) that uses all 8 bits. DCF uses this 8-bit extension.

ATMS-II. Advanced Text Management System.

ATMS-III. Advanced Text Management System.

attribute. A characteristic of a document (or document element) other than its type or content. For example, the security level of a document or the depth of a figure.

attribute label. In GML markup, a name of an attribute that is entered in the source document when specifying the attribute's value.

B

back matter. In a book, those sections such as glossary or an index that are placed after the main chapters or sections.

balancing. In multicolumn formatting, the process of making column depths on a page approximately equal by redistributing the text in the columns. See also *vertical justification*.

bar code. A code representing characters by sets of parallel bars of varying thickness and separation that are read optically by transverse scanning.

baseline. In a font, the imaginary line on which the bottom of each successive character is aligned.

basic document element. In a general document, one of a group of elements that occurs frequently; for example: note, paragraph, and definition list.

batch environment. An environment in which noninteractive programs are executed.

binding edge. The edge of a page to be bound, stapled, or drilled. Defined with the BIND option of the SCRIPT command.

body. Of a printed page, the portion between the top and bottom margins that contains the text.

boldface. A heavy-faced type. Also, printing in this type.

bottom margin. On a page, the space between the body or the running footing, if any, and the bottom edge of the page.

%%BoundingBox: comment. An Adobe Document Structuring Conventions comment that contains integers describing the coordinates of the lower-left and upper-right corners of the bounding box in the default user coordinate system. The %%BoundingBox: comment coordinates are used by SCRIPT/VS to place the PostScript image in the space reserved within a DCF document.

bounding box coordinates. The coordinates of the lower-left and upper-right corners of an imaginary box surrounding an image in the default user coordinate system. The bounding box coordinates are used by SCRIPT/VS to place a PostScript image in the space reserved within a DCF document.

break. An interruption in the formatting of input lines so that the next input line is printed on a new output line.

bug. An error in a program or in a document markup.

C

call. To transfer control to a procedure, program, routine, or subroutine.

camera-ready copy. Copy which is ready for photographic typesetting.

caps. Capital letters. See also *initial caps*.

caption. Text accompanying and describing an illustration.

case-sensitive. The relevance of a group of letters is uppercase or lowercase. ABC is different from Abc, which is different from ABc.

CDPF. Composed Document Printing Facility.

cell. A single unit within a table, in which text or other expressions may appear. A cell is always rectangular and is usually bounded by horizontal and vertical rules.

centimeter (cm). A measurement equal to one hundredth of a meter; 0.39 inch.

character. A member of a set of elements that is used for the representation, organization, or control of data. Characters may be letters, digits, punctuation marks, or other symbols.

character arrangement table. An array of data that translates input data into printable characters and identifies associated character sets and graphic character modification modules.

character printer. (1) The identification of characters by automatic means. (I) (A) (2) The identification of geographic, phonic, or other characters by various means, including magnetic, optical, or mechanical means. (A)

character set. A finite set of different characters that is complete for a given purpose. For example, in printing, the characters that constitute a font.

character space. The horizontal space of a character. This size depends on the character font and the device on which the character is printed.

character spacing. The space between characters in a word.

cicero. In the Didot point system, a unit of 0.1776 inch (4.512 millimeters) used in measuring typographical material.

CMS. Conversational monitor system—an interactive processor that operates within VM/370.

code page. A font library member that gives the associated code points and character identifiers.

code point. A 1-byte code representing one of 256 potential characters.

coded font. (1) The combination of a code page and a font library. (2) A font library member that is fully described in terms of typeface, point-size, weight, width, and attribute.

color separation. The process of making separate masters of a document for color printing.

column. A vertical arrangement of characters or other expressions.

column balancing. The process of redistributing lines of text among a set of columns so that the amount of text in each column is as equal as possible.

column width. The width of each text column on a page. Specified with the .CW [Column Width] control word. (In multicolumn formatting, all columns on a page usually have the same width.)

command. A request from a terminal or a request specified in a batch processing job for the performance of an operation or the execution of a particular program. For example, a request given at a terminal for SCRIPT/VS to format a document or for an editor to edit a line of text.

comment. A control word line that is ignored by SCRIPT/VS. Such lines begin with either .* or .cm.

composed text. Text that has been formatted and that contains control information to direct the presentation of the text on page printers.

composite. The act or result of formatting a document.

composite rotation. The total amount of rotation done by the printer to place text in the correct orientation on the page.

composition. The act or result of formatting a document.

compositor. A person or program that composes text.

concatenation. An operation that forms an output line containing as many words as the column width allows, by placing the first words from an input line after the last words from the preceding input line. When words from an input line would reach beyond the right margin and hyphenation cannot be performed, they are placed at the beginning of the next output line, and so on.

control word. An instruction within a document that identifies its parts or indicates to SCRIPT/VS how to format the document. See also *macro*.

control word line. An input line that contains at least one control word.

control word statement. A control word and its parameters.

Conversational Monitor System (CMS). A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities, and operates only under control of the VM/370 VM control program.

copy group. A portion of a form definition that defines a set of modifications that can be used when printing a page.

current left margin. The left limit of a column that is in effect for formatting. Each column's left margin is specified with the .CD [Column Definition] control word. However, the current left margin (that is, the left boundary for an output line) might vary to the right of the column's left margin when indentation is changed with the .IN [Indent], .UN [Undent], .IL [Indent Line], and .OF [Offset] control words.

current line. The line in a source document at which a computer program (such as an editor or a formatter) is positioned for processing.

D

debug. (1) To detect, diagnose, and eliminate errors in computer programs and SCRIPT/VS documents. (T) (2) Synonymous with checkout, troubleshoot.

default value. A value assumed by a computer program when a control word, command, or control statement with no parameters is processed. In GML processing, the value assumed for an attribute when none has been specified.

deferred control words. SCRIPT/VS control words that are processed after the text has been placed on the page.

descender. (1) In a font, the distance from the baseline to the bottom of the character. See *maximum descender*. (2) The part of a letter that falls below the body of the letter. Letters with descenders are g, j, p, q, y, and Q.

destination. (1) Any point or location, such as a node, station, or particular terminal, to which information is to be sent. (2) An external logical unit LU or application program to which messages or other data are directed.

dictionary. A collection of *word stems* that is used with the spelling verification and automatic hyphenation functions.

Didot point system. A standard printer's measurement system on which type sizes are based. A Didot point is 0.0148 inch (0.376 millimeter). There are 12 Didot points to a cicero. See also *cicero* and *point*.

document. A publication or other written material. See also *output document* and *source document*.

document administrator. A person who is responsible for defining markup conventions and procedures for an organization.

document conversion processor. A computer program that processes a machine-readable document that includes formatting controls written in one formatter language, to produce a machine-readable document that includes formatting controls appropriate for another formatter language.

document library. A set of VSAM data sets, accessible in a batch environment, that contain documents and related files.

dot leader. A set of periods that fills in the space between two pieces of split text such as a chapter title and its page number in a table of contents.

download. To transfer data from a processing unit to an attached device such as a microcomputer for processing. Contrast with *upload*.

duplex. A mode of copying or printing on both sides of a sheet.

E

EBCDIC. Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

edit. To add, change, delete, or rearrange data and to perform operations such as code conversion and zero suppression.

editor. (1) See *linkage editor*. (2) See *editor program*.

editor program. (1) A computer program designed to perform such functions as rearrangement, modification, and deletion of data in accordance with prescribed rules. (2) Contrast with *linkage editor*.

eject. In formatting, a skip to the next column or page.

element. Any part of a document: a single character, a word, or a sentence. Also refers to any part of a document you can identify with a GML tag (tagged element), such as a paragraph, figure, or heading.

em. A unit of measure equal to the width or the height of the character "m" in a particular font.

en. A unit of measure usually equal to one-half the width of an em. For many typefaces, the average width of lower case characters tends to be equal to the width of an en.

enabled. Used in reference to a tag, it means that the tag is mapped to its appropriate APF.

Encapsulated PostScript. Any file containing the PostScript Page Description Language that conforms to Adobe 2.0 Document Structuring Conventions and follows conventions defined by Adobe Systems to allow the file to be included by other applications. Encapsulated PostScript can be included in a DCF document by means of the .PO [PostScript] control word.

epifile. The portion of a profile (after a .EF control word) that is processed *after* the main document has been processed.

escapement. (1) Movement of one character space between the paper carrier and typing or printing position, parallel with the typing or printing line. (2) The unit of vertical or horizontal movement that is built into a device. For the 4250, that value is 1/600th of an inch; for the 3800 Model 3 and the 3820 Page Printer, that value is 1/240th of an inch; and for PostScript devices, that value is 1/72000th of an inch.

exposure. The amount of risk associated with a schedule item or items.

extended symbol processing. The processing of a symbol whose value causes the rest of the line to be stacked and later processed as a new input line.

F

factor. A dimensionless scalar value used to form a product with another value. Factors can also be expressed as percentages.

FBA. Fixed-block-architecture.

figure space. (1) A unit of measure equal to the width of the “en” space in a particular font. (2) In the Document Composition Facility, the width of the figure (0).

fill character. A character used to occupy a space; for example, blanks used to fill up the space left by tabbing.

float. (1) A keep (group of input lines kept together) whose location in the source file can vary from its location in the printed document. (2) Of a keep, to be formatted in a location different from its location in the source file.

flush. Having no indentation.

fold. (1) To translate the lowercase characters of a character string into uppercase. (2) To place that portion of a line that does not fit within a column on the next output line.

folio. Page number.

font. A font library member that contains characters that must be used in conjunction with a code page font library member.

font object. A member of a font library. In CMS, a file whose filetype matches the name of the font library. In MVS, a member of a partitioned data set (PDS).

font set. The set of fonts to be used in formatting a source document.

footer. Text that appears at the bottom of every page of a document, for example, a page number.

footing. Words located at the bottom of the text area. See also *running footing*.

footnote. A note of reference, explanation, or comment, placed below the text of a column or page, but within the body of the page above the running footing.

foreground. The environment in which interactive programs are executed. Interactive processors reside in the foreground.

format. (1) The shape, size, and general makeup of a printed document. (2) To prepare a document for printing in a specified format.

formatter. (1) A computer program that prepares a source document for printing. (2) That part of SCRIPT/VS that formats input lines for a particular type of logical device.

formatting mode. In document formatting, the state in which input lines are concatenated and the resulting output lines are justified.

FORMDEF. Synonym for form definition.

form definition (FORMDEF). A resource object that defines the characteristics of the form which include: overlays to be used, text suppression, position of page data on the form, and number and modifications of a page. Synonymous with FORMDEF.

front matter. In a book, those sections (such as preface, abstract, table of contents, list of illustrations) that are placed before the main chapters or sections.

G

general document. A type of document whose description can apply to a variety of documents, from memoranda to technical manuals. It can be used as a catch-all category for documents that do not conform to any other type description.

Generalized Markup Language (GML). A language that can be used to identify the parts of a source document without respect to a particular processing system.

GML delimiter. A special character that denotes the start of GML markup. In the starter set, it is initially a colon (:).

GML end tag delimiter. A special character that denotes the end of GML markup. In the starter set, it is initially a period (.).

GML interpretation. Recognizing the start or end of an element (or an attribute label), associating it with an APF, and executing the APF. In SCRIPT/VS, interpretation is performed jointly by SCRIPT/VS and by APFs.

graphic character modification (GCM) module. Modules that contain scan patterns of IBM-supplied character sets, user-defined character sets, or both without respect to particular processing.

graphical data display manager (GDDM). An IBM licensed program that creates page segments.

gutter. In multicolumn formatting, the space between columns.

H

hanging indentation. The indentation of all lines of a block of text following the first line, which is not indented the same number of spaces. Specified with the .OF [Offset] or .UN [Undent] control word.

heading. Words located at the beginning of a chapter or section or at the top of a page, above the first line of text on the page. See also *head-level* and *running heading*.

heading segment. An element that begins with a heading, followed by basic document elements and lower-level heading segments.

head-level. The typeface and character size associated with the words standing at the beginning of a chapter or chapter topic.

hexadecimal. Pertaining to a numbering system based on 16, using the sixteen digits 0, 1, . . . 9, A, B,

C, D, E, and F. For example, hexadecimal 1B equals decimal 27. See also *EBCDIC*.

highlighting. Emphasis associated with a document element. In formatting, highlighting is usually expressed by changing a font, overstriking, under-scoring, or capitalizing the highlighted element.

horizontal justification. The redistribution of horizontal white space at the end of a line of text to the spaces between the words and letters of the line in order to exactly fill the width of the column with the text.

I

IEBIMAGE. A utility program that creates and maintains various 3800 Printing Subsystem Model 1 modules (for example, character arrangement table and graphic character modification (modules) and stores them in SYS1.IMAGELIB.

image. (1) A likeness or imitation of an object, such as a picture or logo. (2) A PostScript file that can contain any combination of images and text.

impact printer. A printer, in which printing is the result of mechanical impacts. (I) (A)

implied paragraph structure. An element that begins with an implied paragraph; that is, one for which you do not specifically enter a paragraph tag. The existence of the paragraph is understood from the existence of the implied paragraph structure, for example, as with notes, figure captions, and lists.

indent. To set typographical material to the right of the left margin, while still retaining the original (fixed) margin settings.

indentation. The action of indenting. The condition of being indented. The blank space produced by indenting. Specified with the .IN [Indent], .IR [Indent Right], .UN [Undent], .OF [Offset], and .IL [Indent Line] control words. See also *hanging indentation*.

initial value. A value assumed by SCRIPT/VS for a formatting function until the value is explicitly changed with a control word. The *initial value* is assumed even before the control word is encountered, whereas the *default value* is assumed when the control word is issued without parameters. See also *default value*.

initialize. (1) In programming languages, to give a value to a data object at the beginning of its lifetime. (2) To set counters, switches, addresses, or contents of storage to zero or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

inline space. The amount of horizontal white space that usually occurs between words in a line.

input device. Synonym for input unit.

input line. A line, as entered into a source file, to be processed by a formatter.

input unit. A device in a data processing system by means of which data can be entered into the system. (I) (A) Synonymous with input device.

interactive. Pertaining to an application in which entries call forth a response from a system or program, as in an inquiry system. An interactive system might also be conversational, implying a continuous dialog between user and system. Interactive systems are usually communicated with via terminals, and they respond to commands. See also *foreground*.

interactive environment. The environment in which an interactive processor operates.

Interactive System Productivity Facility (ISPF). A dialog manager for interactive applications that provides control and services to allow processing of the dialogs in different host environments.

intercharacter space. Extra horizontal white space inserted *between* characters of a word. This space is in addition to the space included as part of the characters by the designer of the font.

interword space. See *word space*.

ISPF. Interactive System Productivity Facility.

italic. A type style with characters that slant to the right.

J

job control language (JCL). A control language used to identify a job or describe its requirements to the operating system.

job control statement (JCS). A statement in a job that is used in identifying the job or describing its requirements to the operating system.

justification. See *horizontal justification* and *vertical justification*.

justify. To control the printing positions of characters on a page so that the left-hand and right-hand margins of the printing are regular. (I) (A) See *left-justify* and *right-justify*.

K

kanji. The nonphonetic Japanese writing system. In a font representing kanji characters, each character is represented by a double-byte code. Contrast with *katakana*.

katakana. A character set consisting of symbols used in one of the two common Japanese phonetic alphabets. Each character is represented by one byte. Contrast with *kanji*.

keep. In a source document, a collection of lines of text to be printed in the same column. When the vertical space remaining in the current column is insufficient for the block of text, the text is printed in the next column. In the case of single-column format, the next column is on the next page.

L

layout. The arrangement of matter to be printed. See also *format*.

leader. (1) Dots or hyphens (as in a table of contents) used to lead the eye horizontally. (2) The divider between text and footnotes on a page, usually a short line of dashes.

left-hand page. The page on the left when a book is opened; usually even-numbered.

left-justify. To control the printing positions of characters on a page so that the left-hand margin of the printing is regular. (I) (A)

legend. An explanatory list of the symbols, lines, and other components of a schedule.

ligature. A single character (piece of type or font raster) that represents two or more input characters: ff and ffi are examples of characters that may be represented by (printed as) a ligature.

line device. Any of a class of printers that accept one line of text from the host system at a time. SCRIPT/VS supports such line devices as the 1403 Printer and 3800 Model 1.

line printer. (1) A device that prints a line of characters as a unit. (I) (A) (2) Contrast with character printer, page printer.

line space. The vertical distance between the baseline of the current line and the baseline of the preceding line.

line spacing. See *line space*.

logical output device. The combination of a physical output device and such logical variables as page size

and number of lines per vertical inch (for line devices). A specification of 1403W6 is an example of a logical output device.

logical page. Synonym for page.

lowercase. Pertaining to small letters as distinguished from capitals, for example: a, b, g rather than A, B, G.

M

machine-readable. Pertaining to data a machine can acquire or interpret (read) from a storage device, or a data medium, or other source.

maclib. See *macro library*

MACLIB library. A library that contains macros, copy files, or source program statements for use under CMS.

macro. See *macro instruction*

macro instruction. (1) An instruction that when executed causes the execution of a predefined sequence of instructions in the same source language. In SCRIPT/VS, a macro is a sequence of one or more control words, symbols, and input lines. A macro's definition can be recursive. (2) Synonymous with macrostatement.

macro library. A library of macro definitions used during macro expansion. The form the library takes will vary by environment, being a MACLIB in CMS, a PDS in TSO, and so on.

macrostatement. Synonym for macro instruction.

macro substitution. During formatting, the substitution of control words, symbols, and text for a macro.

map. To associate a tag with an APF, using the .AA [Associate APF] control word.

margin. (1) The space above, below, and on either side of the body of a page. (2) The left or right limit of a column.

mark up. (1) To determine the markup for a document. (2) To insert markup into a source document.

markup. Information added to a document that enables a person or system to process it. Markup information can describe the document's characteristics, or it can specify the actual processing to be performed. In SCRIPT/VS, markup consists of GML tags, attribute labels and values, and control words.

markup-content separator. A delimiter used in GML markup that indicates the end of the markup and the

beginning of the text. The default markup content separator for GML is a period (.).

maximum ascender. The maximum height from the baseline of sequential characters to the top mark of the tallest character in a font character set.

maximum descender. The maximum depth from the baseline of sequential characters to the bottom mark of any character in the font character set.

MCS. Markup/content separator.

meter (m). 1.0936 yards; 3.2808 feet; 39.3696 inches.

millimeter (mm). One thousandth of a meter; 0.04 inch.

N

National Language Support. Translation requirements affecting parts of devices and licensed programs; for example, rules for translation of message text, and for conversion of symbols such as the US dollar sign to the UK pound sign.

nonimpact printer. A printer, such as the 3800 Printing Subsystem, in which printing is not the result of mechanical impacts, but is instead produced by another process such as laser beam, ink jet, or electroerosion. The 3800 Printing Subsystem, for example, uses a laser based technology, and the 4250 Printer uses an electroerosion process.

O

object. A sequential collection of control records that represents documents, pages, fonts, and so on.

offset. (1) To indent all lines of a block of text, except the first line. (2) The indentation of all lines of a block of text following the first line.

option. Information entered with a SCRIPT command to control the execution of SCRIPT/VS.

orientation. In the 3800 Printing Subsystem Models 3 and 8, the number of degrees an object is rotated relative to a reference; for example, the orientation of printing on a page, relative to the page coordinates. See also *text orientation*.

output device. Synonym for output unit.

output document. A machine-readable collection of lines of text or images that have been formatted, or otherwise processed, by a document processor. The output document can be printed or it can be filed for future processing.

output line. A line of text produced by a formatter.

output unit. A device in a data processing system by which data can be received from the system. (I) (A) Synonymous with output device.

overlay. A collection of predefined data such as lines, shading, text, boxes, or logos that can be merged with variable data on a page while printing.

P

page. A collection of data that can be printed on a physical sheet of paper. Synonymous with logical page.

PAGEDEF. Page definition.

page definition (PAGEDEF). (1) A resource, specified in the print data set JCL, that defines the rules for transforming the input to pages and text controls. (2) In the 3800 Print Management Facility, a member of a partitioned data set that contains the formatting instructions for a print data set, although it can be used for any compatible print data set.

page device. A device that prints a formatted page that has graphics and text merged.

page printer. (1) Any of a class of devices that accept composed pages, constructed of composed text and images, and prints one page as a unit. SCRIPT/VS supports such page printers as the 4250 Printer, the 3800 Model 3, and the 3820 Page Printer. (2) Contrast with *character printer, line printer*.

page segment. (1) An object that can contain text and images and be included on any addressable point on a page or electronic overlay. It assumes the environment of the object it is included in. (2) A library member that contains the definition of the page segment.

paginate. To number pages.

paragraph unit. An element that has the same structure as a paragraph. In a general document, the paragraph units are: paragraph, note, and paragraph continuation.

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (I) (A) The syntax of some SCRIPT/VS control words includes parameters, which establish the properties of a formatting function or a printed page. (2) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted.

part. In a general document, a part is a zero-level heading segment. See also *heading segment*.

patch PSC element. A PSC element that is used temporarily to modify the normal output.

pel. (1) An element of a raster pattern about which a toned area on a photoconductor can appear. See also *raster pattern*. (2) The unit of horizontal measurement for the 3800 Printing Subsystem and the 4250 Printer. On the 3800 Printing Subsystem Model 1, one pel equals approximately 1/180th inch. On the 3800 Model 3 and the 3820 Page Printer, one pel equals approximately 1/240th inch. On the 4250 Printer, one pel equals approximately 1/600th inch.

pel density. The number of picture elements per inch of linear measurement.

physical output device. A physical device, that stores, prints, or displays data, such as a terminal, a disk file, a line printer, or a nonimpact printer. The 1403 is an example of a physical output device.

pica. A unit of about 1/6 inch used in measuring typographical material. It is similar to a cicero in the Didot point system.

pitch. On a typewriter, the distance between corresponding points of two equal characters that are typed immediately adjacent to one another.

point. (1) A unit of about 1/72 of an inch used in measuring typographical material. There are 12 points to the pica. (2) In the Didot point system, a unit of 0.0147 inch. There are 12 Didot points to the cicero.

PostScript devices. Any of a class of devices that accept the PostScript page description language, such as the IBM 4019 LaserPrinter. When used with DCF, PostScript devices must be configured to accept 8-bit ASCII.

PostScript language. A programming language designed to convey a description of virtually any desired page to a printer. It possesses a wide range of graphic operators that may be combined in any manner.

PostScript image file. Any file containing encapsulated PostScript that is imbedded in a DCF document by means of the .PO [PostScript] control word. PostScript image files can include any combination of images or text.

profile. (1) In SCRIPT/VS processing, a file that is imbedded before the primary file is processed. It can be used to control the formatting of a class of source documents. When processing GML markup, the profile usually contains the mapping from GML to APFs and the symbol settings that define the formatting style. (2) In the DLF library, a collection of information that identifies a batch SCRIPT/VS user (user profile) or a document processor (attribute profile) or that defines certain library parameters (system profile).

proportional spacing. The spacing of characters in a printed line so that each character is allotted a space proportional to the character's width.

R

ragged right. Pertaining to text that is not right-justified. See also *left-justify*.

ragged left. Pertaining to text that is not left-justified. See also *right-justify*.

reference element. In a general document, an element whose content is a reference to another element that is generated by an APF. There are five: figure reference, footnote reference, heading reference, index entry reference, and list item reference.

required blank. A character that prints as a blank, but does not act as a word separator.

required space. In word processing, a space or blank that must not be removed when adjusting a line or paragraph of text. A character that prints as a blank, but does not act as a word separator. Synonym for required blank.

residual text. The line of text following the markup/content separator of a GML tag.

RFTDCA. Revisable-Form-Text Document Content Architecture specifies how IBM office systems interchange documents that are in revisable form. This architecture defines the structure of the data streams that represent revisable-form-text documents within the office system or network.

right-hand page. The page on the right when a book is opened; usually odd-numbered.

right-justify. To control the positions of characters on a page so that the right-hand margin of the printing is regular. (I) (A)

row. A horizontal arrangement of characters or other expressions on a printed page.

rule. (1) A straight horizontal or vertical line used, for example, to separate or border the parts of a figure or box. (2) A solid or patterned line of any weight, extending horizontally across the row or vertically down the column.

running footing. A footing that is repeated above the bottom margin area on consecutive pages or on consecutive odd-numbered or even-numbered pages in the text area of the page. Synonymous with footer.

running heading. A heading that is repeated below the top margin area on consecutive pages or on consecutive odd-numbered or even-numbered pages in the text area of the page.

S

SCRIPT/VS. The formatter component of the Document Composition Facility. SCRIPT/VS provides capabilities for text formatting and document management, macro processing and symbol substitution, and GML tag recognition and processing.

section. Each part of the output page when an output page has two or more single-column parts with the same or different column-widths, or a single-column part and a multicolumn part, or two or more different multicolumn parts.

segment. An object containing composed text and images, prepared before formatting and included in a document when it is printed.

separation masters. The process of making separate masters of a document for the purpose of special printing (such as multi-part forms or multiple-color printing).

set. Used in reference to a symbol; it implies the .SE [Set Symbol] control word.

set size. The set size of a given typeface determines the number of characters that will fit in a line of a given width when it is printed or set.

shading. Highlighting an area on the page by varying graded density.

showpage. A PostScript command that transmits the current page to the current output device, causing any marks on the page to actually appear.

slip end date. The end date of a schedule or project that has moved or *slipped* to a later time than originally planned.

slip start date. The start date of a schedule or project that has moved or *slipped* to a later time than originally planned.

small caps. Capital letters in the same style as the normal capital letters in a font, but approximately the size of the lowercase letters.

source document. A machine-readable collection of lines of text or images that is used for input to a computer program.

space. A blank area separating words or lines.

space unit. A unit of measure of horizontal or vertical space. In GML markup, the em is used when a measure that is relative to the current font size is required. When an absolute measure is required, as in specifying the depth of a figure, recommended space units are inches (nnI), millimeters (nnW),

picas/points (nnPnn), or Ciceros/Didot points (nnCnn), where nn is the number of units. See also *em*, *pica*, *point*, *Cicero*, and *Didot point system*.

starter set. An example of GML support that is provided with the Document Composition Facility. It consists of a document-type description for general documents, a profile, and a library of APFs.

string. A sequence of elements of the same nature, such as characters considered as a whole.

structure. A characteristic of a document (or element) that expresses the type and relationship of the elements of the content. See also *content* and *element*.

structured field. A self-identifying string of bytes, analogous to a logical record. A structured field consists of an introducer, which identifies and characterizes the structured field, and data or parameters.

symbol. A name in a source document that can be replaced with something else. In SCRIPT/VS, a symbol is replaced with a character string. SCRIPT/VS can interpret the character string as a number, a character string, a control word, or another symbol.

symbol substitution. During formatting, the replacement of a symbol with a character string that SCRIPT/VS can interpret as a value (numeric, character string, or control word) or as another symbol.

SYSVAR. An option of the SCRIPT command that permits the user to specify values for symbols. In the starter set, SYSVAR symbol values determine whether certain processing variations will occur, such as heading numbering, duplex formatting, and two-column printing.

T

tab. (1) (noun) A preset point in the typing line of a typewriter-like terminal. A preset point in an output line. (2) (noun) A tab character, X'05'. (3) (verb) To advance to a tab for printing or typing.

table. (1) An array of data each item of which can be unambiguously identified by means of one or more arguments. (I) (A) (2) An arrangement of cells in rows and columns.

tag. In GML markup, a name for a type of document or document element that is entered in the source document to identify it. For example, *:p.* is the tag used to identify a paragraph.

terminal. A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a communication channel.

text item. Explicitly marked elements that occur within text, such as within a paragraph unit. In a general document, for example, quotations and phrases are text items.

text orientation. A description of the appearance of text as a combination of print direction and character rotation.

text programmer. One who implements APFs that provide the processing specified by the document administrator. In SCRIPT/VS, this involves writing SCRIPT/VS macros and organizing macro libraries and profile files so that the appropriate composition will be done for each tag.

text line. A line that contains only text.

text variable. A symbol whose final value is to be treated only as text.

time sharing option (TSO). An option on the operating system; for System/370, the option provides interactive time sharing from remote terminals.

token. A string of characters that is treated as a single entity. In SCRIPT/VS, a parameter passed to a macro in one of the local variables &*1, ... &*n.

top margin. On a page, the space between the body or running heading and the top edge of the page.

transparency. A master or copy on material that transmits light without diffusion.

translation table. A table that provides replacement characters for characters that cannot be printed or that substitutes characters from a different font. For example, the 256-byte portion of the character arrangement table that translates the user's data code for a character recognizable by the 3800 Printing Subsystem Model 1.

TRC. Table reference character (TRC) in printer SYSOUT data sets, a second control byte, following the carriage control byte, which indicates which font the record is to be printed in. The presence of TRCs is indicated by the JCL parameter DCB=OPTCD=J.

TSO. Time-sharing option.

typeface. All type of a single style. There might be several fonts (different sizes) with the same typeface or style.

typeface family. A collection of fonts of a common typeface that vary in size and style.

typeset. (1) To arrange the type on a page for printing. (2) Pertaining to material that has been set in type.

type posture. A typeface style variation indicating whether a typeface is upright (as in Roman) or slanted to the right (as in italic or cursive).

type size. A measurement in pitch or points of the height and width of a graphic character in a font. For example, the vertical height (point size) of a given typeface, such as 10 point.

type style. The form of characters within a set of the same font, for example: elite, or pica. (T) Attributes such as posture, weight, and width may vary in a type style.

type weight. (1) The degree of boldness of a typeface series, caused by different thicknesses of the strokes that form a graphic character. (2) One of the many attributes of a font, others, for example: being size and typeface.

type width. The horizontal size (set size) of a given typeface. The width may be given in units of measurement, such as set 9 point, or it may be descriptive: ultra condensed, condensed, expanded, and so on.

U

underscore. (1) A line printed under one or more characters. (2) To place a line under one or more characters; to underline.

unformatted mode. (1) In document formatting, the state in which each input line is processed and printed without formatting. Other SCRIPT/VS control words remain in effect and are recognized. (2) In document printing, using the UNFORMAT option, the state in which each input line (control words as well as text) is printed as it exists in the input, in the order in which it is processed. No formatting is done.

unique identifier (ID). In a general document, an attribute whose value serves as a name that can be used to refer to the element. See also *reference element*.

unit space. The minimum amount of additional spacing acceptable for purposes of horizontal justification, as specified by the font designer.

upload. To transfer data from a device such as a microcomputer to a processing unit. Contrast with *download*.

uppercase. Pertaining to capital letters as distinguished from small letters, for example: A, B, G rather than a, b, g.

V

variable. A quantity that can assume any of a given set of values.

variable text. For the .VT [Variable Text] control word, text to be inserted in a formatted document by a postprocessor.

vertical justification. Redistribution of the extra vertical white space at the end of a column between lines of text, so as to make the columns appear to be the same length.

W

widow. A line or word by itself ending a paragraph.

word space. The white space placed between words in a line. This is sometimes referred to as an inter-word space.

word spacing. The space between words in a line. See also *word space*.

writable character generation module (WCGM). In the 3800 Printing Subsystem, a 64-position portion of character generation storage that holds the scan elements of a single character set. There are two WCGMs in the basic 3800, and optional added storage provides two more.

X

XPO. See *exposure*

Index

Special Characters

- .GDOC tag
 - LANGUAGE attribute 169
- ... [Set Label] 234
 - using 45
- .AA [Associate APF] 74, 234
 - using 79
- .AM [Adjust Margin] 234
- .AN [And] 234
 - using in macros 57
 - using to check multiple conditions 45
 - using with .IF 43
- .AP [Append] 234
- .AR [Area] 234
- .BC [Balance Columns] 234
- .BF [Begin Font] 234
- .BL [Blank Line] 234
- .BM [Bottom Margin] 234
- .BX [Box] 235
- .CC [Conditional Column Begin] 235
- .CD [Column Definition] 235
- .CE [Center] 235
- .CG [Copy Group] 235
- .CL [Column Line Length] 235
- .CP [Conditional Page Eject] 235
- .CS [Conditional Section]
 - IGNORE parameter of 47
 - INCLUDE parameter of 48
 - using 47
- .CT [Continued Text] 235
- .DA [Define Area] 235
- .DC [Define Character] 24, 235
 - ASEP parameter of 32
 - PUNC parameter of 87, 91
 - WORD parameter of 86
- .DD [Define Data File-id] 235
- .DF [Define Font] 191, 195, 201, 235, 261, 266
 - with SEPMASTR keyword 192, 201
- .DH [Define Head Level] 235
- .DL [Dictionary List] 177, 236, 313
 - specifying languages with 89
 - specifying user dictionary with 157
- .DM [Define Macro] 236
 - using inline form of 54
 - using subscripted form of 55
 - using to define macros 54
- .DR [Define Rule] 191, 193, 195, 202, 236
 - with SEPMASTR keyword 192, 193, 202
- .DU [Dictionary Update] 236
 - ADD parameter of 89, 92
 - DEL parameter of 92
 - using 91
- .DV [Define Variable] 37, 236
 - using 36
- .EC [Execute Control] 236
 - using 60
- .EF [End of File] 236
- .EL [Else] 236
 - using for alternative processing 44
 - using in macros 57
 - using with .IF 43
- .EM [Execute Macro] 236
 - using 60
- .EQ [Equation] 236
- .ES [Extra Space] 236
- .FL [Float] 236
- .FN [Footnote] 236
- .FO [Format Mode] 236
- .FV [Format Vertically] 236
- .GO [Goto] 236
 - using to bypass part of a file 45
- .GS [GML Services] 237
 - using 71, 79
- .HN [Headnote] 237
- .HR [Horizontal Rule] 237
- .HY [Hyphenate] 237
 - NOALG parameter of 93
- .IC [Intercharacter Space] 237
- .IE [Index Entry] 237
- .IF [If] 237
 - comparing null values 46
 - resolving symbols 47
 - using for alternative processing 44
 - using for conditional processing 43, 45
 - imbedding a file in several master files 45
 - using for conditional testing 43
 - using in macros 57
 - using SYSOUT comparand with 46
 - using SYSPAGE comparand with 46
 - using with substitution off 43
- .IM [Imbed] 237
- .IN [Indent] 237
- .IR [Indent Right] 237
- .IS [Inline Space] 237
- .IT [Input Trace] 237
- .IX [Index] 238
- .JU [Justify Mode] 238
- .KP [Keep] 238
- .LB [Leading Blank] 63, 238
- .LI [Literal] 238
- .LL [Line Length] 238
- .LO [Lead-Out] 238
- .LS [Line Spacing] 238
- .LT [Leading Tab] 63, 238
- .LY [Library] 238
 - MAC parameter of 65

- .LY [Library] *(continued)*
 - SYM parameter of 65
 - using in an ATMS-III environment 66
- .ME [Macro Exit] 238
- .MG [Message] 238
- .MS [Macro Substitution]
 - required for macro processing 53
- .NF [No Formatting] 238
- .NL [Null Line] 238
- .OC [Output Comment] 238
 - defining for postprocessor use 211
 - formatting effect of 211
- .OF [Offset] 238
- .OI [Overlay Include] 191, 194, 195, 238
 - with SEPMASTR keyword 192, 194
- .OR [Or] 238
 - using in macros 57
 - using to check multiple conditions 45
 - using with .IF 43
- .PA [Page Eject] 239
- .PF [Previous Font] 239
- .PI [Put Index] 239
- .PL [Page Length] 239
- .PM [Page Margins] 239
- .PN [Page Numbering Mode] 239
- .PO [PostScript] 191, 194, 195, 239
 - with SEPMASTR keyword 192, 194
- .PT [Put Table of Contents] 239
- .PW [Page Width] 239
- .PX [Prefix] 239
- .RC [Revision Code] 239
 - defining for postprocessor use 211
 - formatting effect of 211
- .RD [Read Terminal] 239, 300
- .RE [Restore Environment] 239
 - using 18
- .RF [Running Footing] 240
- .RH [Running Heading] 240
- .RI [Right Adjust] 240
- .RN [Reference Numbers] 240
- .RV [Read Variable] 240
- .SA [Save Environment] 240
 - using 18
- .SB [Shift Baseline] 240
- .SD [Shade Definition] 191, 194, 195, 240
 - with SEPMASTR keyword 192, 194
- .SE [Set Symbol] 240
- .SI [Segment Include] 107, 191, 195, 240
 - with SEPMASTR keyword 192, 195
- .SK [Skip] 240
- .SM [Separation Masters] 191, 192, 201, 240
- .SO [STAIRS/VS Output] 241
 - using 208
- .SP [Space] 241
- .SU [Substitute Symbol] 241
- .SV [Spelling Verification] 241
 - NOADD parameter of 88
 - NOSTEM parameter of 88
- .SV [Spelling Verification] *(continued)*
 - NUM parameter of 87
- .SX [Split Text] 241
- .SY [System Command] 27, 217, 241
- .TA [Table] 105, 241
- .TB [Tab Setting] 241
- .TC [Table of Contents] 241
- .TD [Table Definition] 241
- .TH [Then] 241
 - using for alternative processing 44
 - using in macros 57
 - using with .IF 43
- .TI [Translate Input] 241
- .TM [Top Margin] 241
- .TP [Tab Position] 241
- .TS [Translate String] 241
- .TU [Translate Uppercase] 171, 173, 241
- .TY [Type on Terminal] 300
- .UC [Underscore and Capitalize] 242
- .UD [Underscore Definition] 242
- .UN [Undent] 242
- .UP [Uppercase] 242
- .US [Underscore] 242
- .UW [Unverified Word] 242
 - using 85
- .VR [Vertical Rule] 242
- .VT [Variable Text] 242
- .WF [Write To File] 242
 - using to dynamically create input files 209
- .WS [Word Space] 242
- .ZZ [Diagnostic] 217, 242
- &\$ASCII 24
- &\$CW 72
- &\$DCF 24
- &\$DDUT 24
- &\$EGML 24
- &\$ENV 25
- &\$ENVARRAY 25
- &\$GML 25
- &\$IXLG 25
- &\$LC 26
- &\$LDEV 46
- &\$LST 26
- &\$PASS 26
- &\$PDEV 46
- &\$PRT 26
- &\$PSNO 27
- &\$RET 27
- &\$RF 27
- &\$RH 27
- &\$SEPM 28, 191
- &\$SVLG 28, 313, 314
- &\$TAG 28
- &\$TAGD 28
- &\$TWO 27
- &\$VVR 28
- &a' 39

- &AD' 39
- &DH' 39
- &DV' 39
- &E' 39
- &L' 49
- &L' 39
- &r' 39
- &SD' 39
- &SH' 39
- &SV' 39
- &SW' 39
- &T' 39
- &U' 39
- &V' 39
- &W' 39
- &X' 39

Numerics

- 1-to-1 table 147, 181
- 1-to-2 table 147, 181
- 2PASS option 231
- 3800 font GCM contents 248
- 3800 font world trade GCM contents 251
- 3800 Printing Subsystem Model 1 4
- 3800 Printing Subsystem Model 3 4
 - required fonts 151
- 3800 Printing Subsystem Model 6 4
- 3800 Printing Subsystem Models 3 and 6 4
- 3812 Page Printer 4
- 3816 Page Printer 4
- 3820 Page Printer 4
 - required fonts 151
- 3825 Page Printer 4
- 3827 Page Printer 4
- 3828 MICR Page Printer 4
- 3835 Page Printer 4
- 3900 Advanced Function Printer. 4
- 4250 Page Printer 4
- 4250 printer
 - required fonts 151

A

- ABSOLUTE parameter xiii
- accented characters code point assignments 89
- ADD parameter
 - of .DU control word 89, 92
- addenda dictionaries
 - building 91
 - definition of 89
 - using for spelling verification 87
- ADJUST option 231
- ADJUSTnn option 231
- Adobe Font Metrics (AFM) files 277, 281
 - adding 278
 - uploading 278
- Advanced Text Management System-III (ATMS-III)
 - See ATMS-III

- AFM (Adobe Font Metrics) files 277, 281
 - adding 278
 - uploading 278
- AFP page printers
 - 3800 Printing Subsystem Models 3 and 6 4
 - 3812 Page Printer 4
 - 3816 Page Printer 4
 - 3820 Page Printer 4
 - 3825 Page Printer 4
 - 3827 Page Printer 4
 - 3828 MICR Page Printer 4
 - 3835 Page Printer 4
 - 3900 Advanced Function Printer. 4
 - IBM 4224-2xx Page Printer 4
 - IBM 4234 Page Printer 4
 - IBM LaserPrinter 4028 4
 - positioning 288
 - required fonts 151
- algorithmic hyphenation
 - default for English language 121
 - installation instructions 317
 - interface 313
 - languages supported 170
 - parameter list 316, 317
 - WORDHYBF 316
 - WORDHYCH 317
 - WORDHYLF 316
 - WORDHYLG 316
 - WORDHYWS 317
 - Release 4 additions 215
 - routines 313
- alpha dictionary report 157
- ampersands 37
 - using as page number symbol 31
- APFs 76, 207
 - associating attributes 74
 - associating tags 74
 - for GML tags 79
 - modifying 210
 - when using SCRIPT/VS as a preprocessor 210
- API (Application Programming Interface) 291
- appending input files
 - symbols set when files are appended 29
- application processing function (APF) 74
- Application Programming Interface (API) 291
- arrays
 - controlling elements of 32
 - index counter of 34
 - specifying 31, 33
- ASCII 24, 278
 - configuring PostScript printer for 8-bit 286
- ASEP parameter 31
 - of .DC control word 32
- associative memory 61
- ATMS-III (Advanced Text Management System-III) 291
 - creating macro libraries in 67
 - formatting 108

ATMS-III (Advanced Text Management System-III)

(continued)

SPA 108

SPP 108

STA 108

STO 108

TSP 108

using .LY control word with 66

attributes

for GML tags

processing 76

rules for 75

symbol 39

&a' 39

&AD' 39

&DH' 39

&DV' 39

&E' 39

&L' 39

&r' 39

&SD' 39

&SH' 39

&SV' 39

&SW' 39

&T' 39

&U' 39

&V' 39

&W' 39

&X' 39

tags 72

B

BALR interface 293

baseline-increment

for PostScript devices 134

BIND option 231

blanks

processing input lines that begin with 63

required 86

BookManager BUILD and READ 323

BookMaster 323

box character sets 130

branching to macros 59

breaking tables xiv

C

Canadian French 178

Canadian French prefixes 95

Canadian French suffixes 95

CECP 184

CENTER option 231

CENTERnn option 231

changes with Release 4.0 xiii

characters

accented 171

code points 184

hex code point assignments 89

characters (continued)

sorting in index 170

special

CHARS option 231, 261

CICS

calling protocol 292

considerations

device type determination 303

terminal output 303

font library support 305

generation 304

GML macros

accessible to the host 304

GML profile documents

accessible to the host 304

host

cleanup 303

Host Command 291

Host Command Check 291

host initialization 293

message 291

page segments

library support 305

program interface 291

SCRIPT/VS

cleanup 302

formatting 296

service routines 291, 296

using with SCRIPT/VS 9

CICS RETURN 292

CICS/VS 9

CMS

creating macro libraries in 66

definition of 9

using MACLIB command 66

COB (Command Options Block) 309

COBFLNAM 295

COBFNAME 295

COBFNUM 295

COBLNAME 296

COBLNUM 296

COBPSNAM 296

COBSNAME 296

COBSNUM 296

COBSW4 298

code pages 152, 184

changing

code page 152

font 152

changing default 152

for PostScript 279

for PostScript devices 134

PostScript 277, 279, 281

modifying 280

code points

accented characters 89

hexadecimal for accented characters 89

translation tables 144

- CODEPAGE parameter 261
- color
 - separations 189
- Command Options Block (COB) 309
- compatibility
 - with earlier releases of SCRIPT/VS 215
 - with SCRIPT/370 215
 - with SCRIPT/VS 3 215
 - with SCRIPT/VS 3.2 215
 - with SCRIPT/VS Release 1 215
 - with SCRIPT/VS Release 2 215
 - with TSO/FORMAT 242
- Condensed Text Format (CTF) 207
 - records 209
- conditional processing
 - special techniques for 46
 - using macros for 57
- conditional section number 47
- configuring PostScript printer for 8-bit ASCII 286
- control card
 - required by dictionary maintenance process 157
- control word separator 37
 - starting a symbol with 35
- control words
 - ... [Set Label] 234
 - .AA [Associate APF] 74, 234
 - .AM [Adjust Margin] 234
 - .AN [And] 234
 - .AP [Append] 234
 - .AR [Area] 234
 - .BC [Balance Columns] 234
 - .BF [Begin Font] 234
 - .BL [Blank Line] 234
 - .BM [Bottom Margin] 234
 - .BX [Box] 235
 - .CC [Conditional Column Begin] 235
 - .CD [Column Definition] 235
 - .CE [Center] 235
 - .CG [Copy Group] 235
 - .CL [Column Line Length] 235
 - .CP [Conditional Page Eject] 235
 - .CT [Continued Text] 235
 - .DA [Define Area] 235
 - .DC [Define Character] 31, 235
 - ASEP parameter 31
 - .DD [Define Data File-id] 235
 - .DF [Define Font] 191, 195, 235, 261, 266
 - and DCFINDEX 261
 - CODEPAGE parameter 261
 - FONT parameter 261
 - TYPE option 266
 - TYPE parameter 261
 - with SEPMASTR keyword 192, 201
 - .DH [Define Head Level] 235
 - .DL [Dictionary List] 28, 177, 236, 313
 - .DM [Define Macro] 236
 - .DR [Define Rule] 191, 195, 236
 - with SEPMASTR keyword 192, 193, 202
- control words (*continued*)
 - .DU [Dictionary Update] 236
 - .DV [Define Variable] 37, 236
 - .EC [Execute Control] 236
 - .EF [End of File] 236
 - .EL [Else] 236
 - .EM [Execute Macro] 236
 - .EQ [Equation] 236
 - .ES [Extra Space] 236
 - .FL [Float] 236
 - .FN [Footnote] 236
 - .FO [Format Mode] 236
 - .FV [Format Vertically] 236
 - .GO [Goto] 59, 236
 - .GS [GML Services] 237
 - .HN [Headnote] 237
 - .HR [Horizontal Rule] 237
 - .HY [Hyphenate] 237
 - .IC [Intercharacter Space] 237
 - .IE [Index Entry] 237
 - .IF [If] 237
 - .IM [Imbed] 237
 - .IN [Indent] 237
 - .IR [Indent Right] 237
 - .IS [Inline Space] 237
 - .IT [Input Trace] 237
 - .IX [Index] 238
 - .JU [Justify Mode] 238
 - .KP [Keep] 238
 - .LB [Leading Blank] 63, 238
 - .LI [Literal] 238
 - .LL [Line Length] 238
 - .LO [Lead-Out] 238
 - .LS [Line Spacing] 238
 - .LT [Leading Tab] 63, 238
 - .LY [Library] 238
 - .ME [Macro Exit] 238
 - .MG [Message] 238
 - .NF [No Formatting] 238
 - .NL [Null Line] 238
 - .OC [Output Comment] 238
 - .OF [Offset] 238
 - .OI [Overlay Include] 191, 195, 238
 - with SEPMASTR keyword 192, 194
 - .OR [Or] 238
 - .PA [Page Eject] 239
 - .PF [Previous Font] 239
 - .PI [Put Index] 239
 - .PL [Page Length] 239
 - .PM [Page Margins] 239
 - .PN [Page Numbering Mode] 239
 - .PO [PostScript] 191, 195, 239
 - with SEPMASTR keyword 192, 194
 - .PT [Put Table of Contents] 239
 - .PW [Page Width] 239
 - .PX [Prefix] 239
 - .RC [Revision Code] 239
 - .RD [Read Terminal] 239, 300

control words (*continued*)

- .RE [Restore Environment] 239
- .RF [Running Footing] 240
- .RH [Running Heading] 240
- .RI [Right Adjust] 240
- .RN [Reference Numbers] 240
- .RV [Read Variable] 240, 300
- .SA [Save Environment] 240
- .SB [Shift Baseline] 240
- .SD [Shade Definition] 191, 195, 240
 - with SEPMASTR keyword 192, 194
- .SE [Set Symbol] 240
- .SI [Segment Include] 107, 191, 195, 240
 - MAP parameter 107
 - with SEPMASTR keyword 192, 195
- .SK [Skip] 240
- .SM [Separation Masters] 191, 192, 240
- .SO [STAIRS/VS Output] 241
- .SP [Space] 241
- .SU [Substitute Symbol] 241
- .SV [Spelling Verification] 241
- .SX [Split Text] 241
- .SY [System Command] 27, 217, 241
- .TA [Table] 105, 241
 - BREAK parameter 105
- .TB [Tab Setting] 241
- .TC [Table of Contents] 241
- .TD [Table Definition] 241
- .TE [Terminal Input] 300
- .TH [Then] 241
- .TI [Translate Input] 241
- .TM [Top Margin] 241
- .TP [Tab Position] 241
- .TS [Translate String] 241
- .TU [Translate Uppercase] 171, 173, 241
- .TY [Type on Terminal] 300
- .UC [Underscore and Capitalize] 242
- .UD [Underscore Definition] 242
- .UN [Undent] 242
- .UP [Uppercase] 242
- .US [Underscore] 242
- .UW [Unverified Word] 242
- .VR [Vertical Rule] 242
- .VT [Variable Text] 242
- .WF [Write To File] 242
- .WS [Word Space] 242
- .ZZ [Diagnostic] 217, 242
- defaults of 10
- direct entry of 211
- redefining 60
- conventions, PostScript output structuring 286
- Conversational Monitor System
 - See CMS
- counter 61
- Country Extended Code Pages (CECP) 184
- courses
 - self-study 325
 - training 325

creating world trade fonts 250

CTF

- option 207, 231
- output 294
- records 209
- current environment
 - saving 18
- current formatting environment
 - &\$ENVARRAY 25
 - displaying levels 25
 - symbol 25
- Customer Information Control System (CICS) 291

D

Danish 178

DCF

- publications
 - library guide 324
- DCFINDEX 261, 263
 - adding entries for PostScript 284
 - and TYPE parameter 261
 - associating the AFM filename 281
 - customizing 277
 - customizing entries for PostScript 285

DDUT option 24, 231

DEBUG option 231

debugging 13

default master 195

defaults

- changing 107, 123, 152
 - NOMAP 107
- coded font 152
- initial work space size 119, 121
- language 119, 120
 - hyphenation 119
 - index sort sequence 119
 - spelling verification 119
- logical device names for output destinations 120, 121
- MVS PROC 161
- values required in environments 119

defining macros 54

defining text variables 36

defining variables 36

DEL parameter

- of .DU control word 92

delimiter characters

- ampersand 37
- control word separator () 37
- GML delimiter (:) 37
- overriding 37
- page number symbol (&) 37
- period 37
- required blank (X'41') 37

DEST option 231

DEVICE option 232

- using to specify STAIRS/VS output 207

- dictionaries
 - addenda 89, 91, 313
 - building 91
 - build process 162
 - changing resident 112
 - hyphenation 170, 314
 - main 157
 - maintenance process 157
 - read-only 89
 - resident 112
 - changing 112
 - default 112
 - root word 89, 170
 - SCRIPT/370 229
 - SCRIPT/VS 89, 93
 - searching order 87
 - spelling verification 170
 - user 89, 91, 157, 160
 - building 91
- dictionary build files 162
- dictionary build process
 - steps of 162
- dictionary maintenance 157
- dictionary maintenance process
 - control card format 159
 - description of 157–165
- Display Editing System 66
- DisplayWrite 4 13
- DisplayWrite/370 13
- DLF (Document Library Facility) 68
- DM PROC 162
- Document Composition Facility
 - Foreground Environment Feature of 9
 - in an interactive environment 9
- Document Library Facility
 - functions of 9
 - using SCRIPT/VS as a subroutine 209
 - using to create SCRIPT/VS macro libraries 68
- DSM#Mxx 171
- DSM#Sxx 171
- DSMACTT 144
- DSMATDEF 124
- DSMATS30 292
- DSMATUOT 112
- DSMBEGP 305
- DSMBOF40 264
- DSMCDP 293
- DSMCICSR 305
- DSMCLOS 297, 302
- DSMCMDEF 126
- DSMCMF40 program 263
- DSMCMUOT 112
- DSMCODE 293
- DSMCSDM 112, 113
- DSMCSPDB 112, 116
- DSMCTFO 298
- DSMDAIXS 185
- DSMDAY 305
- DSMDEF 119
- DSMECDPL 293
- DSMECOB 309
- DSMECOBP 295
- DSMeeUOT 112, 119
- DSMEFCA 297
- DSMEFLG1 298, 301, 302
- DSMEFLG2 297, 298, 299
- DSMEFLS 294
- DSMEFOP 292, 294
- DSMEHCM 294
- DSMEHOST 292, 294
- DSMEICB 291, 306
- DSMEINP 298, 300
- DSMEIOP 292, 294
- DSMELEN 298, 300
- DSMEMOM 294
- DSMEOP 305
- DSMEOUT 294
- DSMEPLS 294
- DSMEREQ 297, 298, 299, 300, 301, 302
- DSMESBEG 292
- DSMESDCO 123
- DSMESLN 298
- DSMESOF 298
- DSMETIME 305
- DSMETIO 294
- DSMFIB 132, 141
- DSMFLIB 301
- DSMFLNF 301, 302
- DSMFLOE 301, 302
- DSMFLUD 301, 302
- DSMHCC 147
- DSMHSTID 294
- DSMHYLIB 215
- DSMIMAP 297
- DSMIXTT1 149, 181
- DSMIXTT2 147, 181
- DSMLDT 131
- DSMLDTP 303
 - CICS services used
 - Interval Control 304
 - Program Control 304
 - Storage Control 304
 - Temporary Storage 304
- DSMLDUOT 112
- DSMLLDEF 127
- DSMLNAM 297, 299, 301, 302
- DSMLOUOT 112
- DSMLPBCS 112, 130
 - DSMBCS 130
 - DSMHCC 131
- DSMLPLDT 112, 131, 152
- DSMLPPDT 113, 136
- DSMMSG 114
- DSMMSGQ 302, 303

- DSMNAME 297, 299, 301, 302
- DSMNEXT 297, 301, 302
- DSMNOIXS 185
 - DSMxxIXS
 - output 185
- DSMNPA 298
- DSMOFMO 305
- DSMOLEN 298, 299, 300
- DSMOPEN 297, 301, 302
- DSMOPRT 298, 305
- DSMOPTR 298, 299, 300
- DSMOPTS 297
- DSMPAG 298, 301, 302
- DSMPDB 119
- DSMPDT 136
- DSMPINP 297
- DSMPLIB 302
- DSMPROF 297
- DSMQNAME 292
- DSMREAS 293
- DSMREOF 301, 302
- DSMRIOE 301, 302
- DSMSABP 305
- DSMSAV1 305
- DSMSAV2 305
- DSMSAV3 305
- DSMSCO 121
- DSMSLDT 136
- DSMSMAX 294
- DSMSMSV 115
- DSMSOUTQ 303
- DSMSPDT 140
- DSMSPEC 297
- DSMSPIXS 185
- DSMSYMAC 297
- DSMTFFIB 113, 140
- DSMTOUTQ 305
- DSMTRC 298
- DSMTRNLT 113, 144, 181
- DSMTSDEF 127
- DSMTSMDM 112
- DSMTSUOT 112
- DSMVXUOT 112
- DSMWDPPT 146
- DSMWRKP 305
- DSMxxIXS 184
 - output 185
- DUMP option 232
- Dutch 177, 178
 - prefixes 96
 - suffixes 96

E

- EBCDIC 24, 278
- English American 178
- English Canadian 178
- English prefixes 94

- English suffixes 94
- English United Kingdom 178
- environment
 - batch 9
 - CICS/VS 9
 - interactive 9
 - MVS 9
 - restoring 19
 - VSE 9
- environment-dependent options tables 119
- even-numbered pages
 - testing for 46
- extended architecture for VM and MVS xiii
- extended symbol processing 35

F

- features, optional
 - Office Document Feature (ODF) 325
 - SCRIPT Mathematical Formula Formatter (SMFF) 325
- figures
 - lists 61
 - referencing 61
- FILE option 232
- file size limitation, removal of xiii
- Finnish 179
- FLIP (font library index program) 261
- FLIP enhancement xiv
- floats
 - formatting environment of 17
- font character set size limitation, removal of xiii
- font library index program (FLIP)
 - concatenation considerations 263
 - content of structured field 262
 - creation of listing 261
 - DCFINDEX member 261
 - operating environments of
 - MVS 264
 - VM/SP CMS 262
 - VSE 265
 - report 266
 - return codes 266
- font library index program (FLIP) enhancement xiv
- FONT parameter 261
- FONTLIB 263
- FONTLIB option 232
- FONTLIBO 263
- fonts
 - 3800 Printing Subsystem Model 1 247
 - provided with SCRIPT/VS 247
 - available for use with SCRIPT/VS 252
 - changing default 152
 - changing in profile 201
 - CICS
 - library support 305
 - for line printers 140
 - Gothic 253
 - Gothic highlight 255

fonts (*continued*)

- Gothic special 257
 - how defaults are chosen 152
 - DSMLPLDT 152
 - GML starter set 152
 - modifying 249
 - PostScript 277
 - creating 278
 - modifying 278
 - PostScript font library 277
 - provided with SCRIPT/VS 252
 - required for
 - 3800 Printing Subsystem Model 3 151
 - 3820 Page Printer 151
 - 4250 printer 151
 - AFP page printers 151
 - IBM 4224 151
 - IBM LaserPrinter 4028 151
 - PostScript devices 153
 - SCRIPT/VS 247
 - list of 247
 - Serif 254
 - Serif highlight 256
 - Serif special 258
- fonts, creating world trade 250
- footnotes
- formatting environment of 17
- Foreground Environment Feature 9
- formatting environment
- defining 17
 - description of 17
 - for footnotes 17
 - for keeps and floats 17
 - for named areas 17
 - for running headings and footings 17
 - named 18
 - parameters of 17
 - saving and restoring 18
- forms
- multiple-part 189
- FPASSES(n) option 26, 27, 105, 232
- improving system performance with 106
- French National 178
- French National prefixes 95
- French National suffixes 95

G

- GCMs (graphic character modification modules) 247, 248, 251
- generic logical device types 4
- German 178
 - prefixes 99
- GML 71
 - delimiter 73
 - defining 73
 - delimiter (:) 37
 - end-tag delimiter 24
 - modifying profiles 201

GML (*continued*)

- profiles
 - DSMMPROF 201
 - DSMPROF4 201
 - DSMSPROF 201
 - DSMTPROF 201
 - modifying 201
 - profiles, modifying 201
 - starter set application 11
 - tag delimiter 25
 - tags
 - :GDOC 169
 - attributes 72
 - automatic processing of 74
 - class mapping 79
 - creating your own 80
 - direct mapping 79
 - explicit mapping 79
 - LANGUAGE attribute of :GDOC 169
 - last one found 28
 - last one processed 28
 - marking up a document with 72
 - markup content separator 72
 - markup syntax 72
 - name requirements 72
 - processing of 79
 - residual text processing 77
 - tag-to-APF processing 79
 - using in macro definitions 53
 - using the .AA [Associate APF] control word 79
 - using the .GS [GML Services] control word 79
 - value attribute processing 77
 - when using SCRIPT/VS as a preprocessor 210
- GML markup
- see GML tags
- Gothic
- highlight text 255
 - special text 257
 - text 253
- graphic character modification modules (GCMs) 247
- contents of 248

H

- header characters, indexes 183
- hexadecimal sort 182
- host operating system, name 119, 120
 - language
 - hyphenation 120
 - index sort sequence 120
 - spelling verification 120
- hyphenation xiii
 - algorithmic 215, 316
 - entry protocol 317
 - exit protocol 317
 - installation instructions 317
 - parameter list 316
 - routine entry and exit protocol 317
 - algorithmic hyphenators 170

- hyphenation (*continued*)
 - algorithmic routine interface 315
 - expanded interface capabilities 315
 - register conventions 315
 - algorithmic routines 313
 - dictionaries 170
 - interface, algorithmic 313
 - prefixes checked for during 93
 - processing 313
 - reports 157
 - search sequence for 93
 - suffixes checked for during 93

I

- IBM 4224
 - required fonts 151
- IBM 4224-2xx Page Printer 4
- IBM 4234 Page Printer 4
- IBM LaserPrinter 4028 4
 - required fonts 151
- ICB (Interface Control Block) 291, 292, 306
- Icelandic 179
- IGNORE parameter
 - of .CS control word 47
- IKJTSMSG 116
- image object content architecture (IOCA) xiii
- imbedding input files
 - symbols set when files are imbedded 29
- INCLUDE parameter
 - of .CS control word 48
- including overlays xiii
- INDEX option 105
 - improving system performance with 106
- indexes
 - counter
 - accessing 34
 - setting 34
 - header characters 183
 - sort key 181
 - sort language 25
 - sort sequence
 - special characters 170
 - tables 169
 - sort sequence macros 147
 - sort sequences 177, 178, 179, 180
 - assigning sort values 182
 - Canadian French 178
 - character code points 184
 - Danish 178
 - Dutch 177, 178
 - English American 178
 - English Canadian 178
 - English United Kingdom 178
 - Finnish 179
 - French National 178
 - German 178
 - Icelandic 179
 - Italian 178
 - modifying 181

- indexes (*continued*)
 - sort sequences (*continued*)
 - modules 184
 - Norwegian 180
 - Portuguese 178
 - Spanish 180
- input files
 - bypassing part of 45
 - conditionally merging 47
 - dynamically creating 209
 - passing parameters to 29
 - preparing for processing 212
- input lines
 - beginning with a blank 63
 - beginning with a tab 63
- input, logically processing 43
- interactive environments
 - using DCF in 9
 - using Foreground Environment Feature in 9
- interface
 - algorithmic hyphenation 313
 - CICS 291
- Interface Control Block (ICB) 291, 292, 306
 - DSMBEGP 305
 - DSMCICSR 305
 - DSMDAY 305
 - DSMEOP 305
 - DSMETIME 305
 - DSMOFMO 305
 - DSMOPRT 305
 - DSMSABP 305
 - DSMSAV1 305
 - DSMSAV2 305
 - DSMSAV3 305
 - DSMTOUTQ 305
 - DSMWRKP 305
- Italian 178
 - prefixes 97
 - suffixes 97

J

- JCL
 - See job control language
- job control language 9

K

- Katakana, processing 173
- keeps
 - formatting environment of 17

L

- labels
 - setting 45
- LADDER 314
- language
 - currently used for hyphenation 28

- language (*continued*)
 - currently used for spelling verification 28
- LANGUAGE attribute 169
- LIB option 232, 296
 - description of 64
 - specifying macro libraries with 65
 - in a CMS environment 66
 - in a TSO environment 67
 - in an ATMS-III environment 68
- library guide
 - for DCF publications 324
- line devices
 - width tables of 143
- line printers
 - font information table 140
- local variables, processing 59
- logical device tables
 - contents 277
 - DSMLDT 131
 - DSMLPLDT 112
- logical devices
 - adding 131
 - definitions 131
- logical output devices
 - definition of 9
 - formatting considerations for 9
 - specifying 9
 - specifying with the DEVICE option 10
- logical processing
 - with symbols 49
- logical processing by SCRIPT/VS
 - methods available for 43
- loops, avoiding 61

M

- MAC parameter
 - of .LY control word 65
- macro
 - libraries
 - list 113
- macros
 - branching 59
 - conditional processing with 57
 - defining 53, 54
 - defining symbols within 57
 - definitions 53
 - DSM#Mxx 171
 - DSM#Sxx 171
 - DSMACTT 144
 - DSMATDEF 124
 - DSMBCS 130
 - DSMCMDEF 126
 - DSMDEF 119
 - DSMECOB 309
 - DSMEICB 291
 - DSMFIB 132, 141
 - DSMHCC 131, 147
 - DSMIXTT1 147, 149, 181

- macros (*continued*)
 - DSMIXTT2 147, 181
 - DSMLDT 131
 - DSMLLDEF 127
 - DSMMSG 114
 - DSMPDB 119
 - DSMPDT 136
 - DSMSCO 121
 - DSMSLDT 136
 - DSMSMSV 115
 - DSMSPDT 140
 - DSMTSDEF 127
 - DSMWDP 146
 - ending 59
 - IKJTSMSG 116
 - libraries 64
 - a batch environment 68
 - in a CMS environment 66
 - in a TSO environment 66
 - in an ATMS-III environment 67
 - specifying 64
 - using 65
 - local symbols for 57
 - naming conventions for 57
 - processing 11, 53
 - redefining SCRIPT/VS control words with 60
 - rules for symbol substitution within 57
 - substituting values for symbols within 56
 - using as associative memory 61
 - using with substitution off 53
 - when to use 53
 - writing 53–68
- main dictionaries
 - searching 157
 - using for spelling verification 87, 157
- managing source documents 211
- map-pseg 140
- mapped page segments 107, 140
- MARK option 232
- MARKUP 323
- markup content separator (MCS) 73
- masters
 - color separation 189
 - default (separation) 195
 - producing separation 196
- maximum number of passes 121
- MAXPT 314
- MCS (markup content separator) 73
 - See *a/so* markup content separator
- memo tags xiv, 201
- memory, associative 61
- MESSAGE option 191, 232
- messages
 - definitions 113, 115
 - translation 171
- MINPT 314
- MINWORD 313

- modification level 24
- module renaming xiv
- modules
 - DSMCSDM 113, 171
 - DSMCSPDB 116, 171
 - DSMDAIXS 185
 - DSMeeUOT 119
 - DSMLPBCS 130
 - DSMLPLDT 131
 - DSMLPPDT 136
 - DSMNIOXS 185
 - DSMSPIXS 185
 - DSMTRNLT 144, 181
 - DSMTSMDM 115, 171
 - DSMxxIXS 184, 185
- multicolumn format
 - processing of for STAIRS/VS output 207
- multiple-color documents 189
- multiple-part forms 189
- MVS
 - running SCRIPT/VS under 9
- MVS and VM extended architecture xiii
- MVS PROC defaults 161

N

- named areas
 - formatting environment of 17
- named formatting environments 18
- National Language Character Definitions 144
- national use considerations 170
 - accented characters 171
 - Katakana characters and symbols 173
 - message translation 171
 - spelling verification and hyphenation
 - dictionaries 170
- NOADD parameter
 - of .SV control word 88
- NOCNT option 232
- NODDUT option 24, 232
- NOMAP 107
- NOPROF option 232
- Norwegian 180
- NOSEGLIB option 232, 305
- NOSORCDD option 123, 232
- NOSTEM parameter
 - of .SV control word 88
- NUM parameter
 - of .SV control word 87
- NUMBER option 232
- NUMBERnn option 232

O

- odd-numbered pages
 - testing for 46
- ODF 325
- ODF (Office Document Feature) 13

- Office Document Feature (ODF) 13, 325
- office systems 13
- OfficeVision 323
- OFFLINE option 232
- oldest usage report 157
- online help xiii
- online help for error messages xiii
- OPEN request 297
- optional features
 - Office Document Feature (ODF) 325
 - SCRIPT Mathematical Formula Formatter (SMFF) 325
- OPTIONS option 232, 297
- output comment control word xiv
- output destination
 - specifying 10
- output structuring conventions, PostScript 286
- overhead transparency tags xiv, 201
- overriding delimiter characters 37

P

- page number symbol limit xiv
- page numbers
 - setting current 31
 - symbol (&) 37
- PAGE option 232
- page segments
 - enhancements xiii
 - library support 305
 - mapped 107, 140
- PAGEnnn option 232
- paper size name
 - for PostScript devices 134
- passes, maximum number 121
- period 37
- phrase definitions 116
- physical device tables
 - adding entries 136
 - DSMLPPDT 113, 136
- physical devices
 - adding 136
 - definitions 136
- physical output devices
 - formatting considerations for 9
 - specifying 9
- point size ranges 266
- pointsize
 - for PostScript devices 134
- Portuguese 178
- postprocessor, using SCRIPT/VS as 209
- PostScript
 - abbreviations, commands 287
 - Adobe Font Metrics files 281
 - AFM files 281
 - baseline-increment 134
 - code pages 134
 - dataset names 280
 - default (PSCP0395) 134
 - list 279

PostScript (*continued*)

- code pages (*continued*)
 - modifying 280
 - sample 281
- command abbreviations 287
- DCFINDEX file 281
 - adding entries 284
 - customizing entries 285
- devices 4
- fonts
 - Adobe Font Metrics files 277
 - code pages 277, 279
 - creating 278
 - DCFINDEX file 277
 - library 277
 - modifying 278
 - required 153
 - support 277
- output structuring conventions 286
- paper size name 134
- pointsize 134
- printers
 - positioning 288
 - unprintable area 287
- producing 277
- space unit notations 279
- uploading AFM file 278
- prefixes
 - Canadian French 95
 - Dutch 96
 - English 94
 - French National 95
 - German 99
 - Italian 97
 - Spanish 101
- prefixes removed during stem processing 93
- preprocessor, using SCRIPT/VS as 210
- PRINT option xiii, 232
- Print Services Facility (PSF) 323
- printer references 4
- printing part of an output document 12
- processing
 - documents with GML (diagram) 75
 - interactively during formatting 12
- producing input for STAIRS/VS 207–209
- products, related
 - See related products
- PROFILE option 232, 297
- profiles
 - when using SCRIPT/VS as a preprocessor 210
- programming interface, CICS 291
- PSF (Print Services Facility) 323
- PSOUT option 24, 232
- publication, related (see related publications)
- publications
 - for DCF 325
- PUNC parameter
 - of .DC control word 87, 91

R

- READ TERMINAL requests 300
- redefining symbols 211
- register conventions 315
 - algorithmic hyphenation routine interface 315
- related products
 - BookManager BUILD and READ 323
 - BookMaster 323
 - MARKUP 323
 - OfficeVision 323
 - Print Services Facility (PSF) 323
 - SGML TextWrite 323
 - SGML Translator DCF edition 323
 - TextTagger ProcessMaster edition 323
 - TextTagger Workstation edition 323
- related publications 323
 - descriptions 326
- Release 4.0 changes
 - breaking tables xiv
 - compatibility with earlier releases 215
 - font library index program (FLIP)
 - enhancement xiv
 - hyphenation xiii
 - IBM LaserPrinter 4028 support xiii
 - including overlays xiii
 - items with name changes 216
 - LANGUAGE attribute on the :GDOC tag xiii
 - memo tags xiv
 - module renaming xiv
 - MVS and VM extended architecture xiii
 - online help xiii
 - online help for error messages xiii
 - output comment control word xiv
 - overhead transparency tags xiv
 - page number symbol limit xiv
 - page segment enhancements
 - ABSOLUTE parameter xiii
 - image object content architecture (IOCA) xiii
 - PRINT option xiii
 - removal of file size limitation xiii
 - removal of font character set size limitation xiii
 - revision character alignment xiv
 - revision code font xiii
 - schedule tags xiv
 - separation masters xiii
 - shading xiii
 - TSO changes xiv
- release level 24
- removal of file size limitation xiii
- removal of font character set size limitation xiii
- reports
 - alpha dictionary 157
 - hyphenation 157
 - oldest usage 157
- required blanks 37, 86
- resident dictionary
 - changing 112

- residual text 73
 - in macro processing 77
- restoring the environment 19
- return codes
 - &\$RET 27
 - font library index program (FLIP) 266
- revision character alignment xiv
- revision code font xiii
- RFT/GML files 13
- RFTDCA files 13
- root word dictionaries 89, 170
 - provided by IBM 89
- root word processing
 - See stem processing
- rules
 - changing in profile 202
- running footing
 - &RF 27
 - depth 27
- running heading
 - &\$RH 27
 - depth 27

S

- sample procedures
 - create and maintain user dictionary 160
- saving the current environment 18
- schedule tags xiv, 201
- SCRIPT command
 - options 230
 - changes to 230
 - setting symbols with 29
- SCRIPT command options
 - 2PASS 231
 - ADJUST 231
 - ADJUSTnn 231
 - BIND 231
 - CENTER 231
 - CENTERnn 231
 - changes 231
 - changing defaults 123
 - CHARS 231, 261
 - CTF 231
 - DDUT 24, 231
 - DEBUG 231
 - DEST 231
 - DEVICE 232
 - DSMEICB 306
 - DUMP 232
 - FILE 232
 - FONTLIB 232
 - FPASSES 105, 232
 - FPASSES(n) 26, 27, 106
 - INDEX 105
 - LIB 232, 296
 - MARK 232
 - MESSAGE 232
 - MESSAGE(DELAY)
 - with SEPMASTR 191

SCRIPT command options (*continued*)

- NOCONT 232
- NODDUT 24, 232
- NOPROF 232
- NOSEGLIB 232, 305
- NOSORCDD 123, 232
- NUMBER 232
- NUMBERnn 232
- OFFLINE 232
- OPTIONS 232, 297
- PAGE 232
- PAGEnnn 232
- PRINT 232
- PROFILE 232, 297
- PSOUT 24, 232
- SEARCH 232, 296
- SEPMASTR 28, 105, 191, 233
 - selecting masters 191
 - with MESSAGE(DELAY) 191
- SINGLE 233
- SORCDD 123, 233
- SPELLCHK 105, 233
- STOP 233
- SYOFF 233
- SYON 233
- SYSVAR 29, 233
- TERM 233
- TLIB 92, 233
- TRANSLATE 233
- TWOPASS 26, 27, 105, 233
- UNFORMAT 233
- UPCASE 127, 233

SCRIPT Mathematical Formula Formatter (SMFF) 325

- program 325

SCRIPT/VS

- 3800 font GCM contents 248
- 3800 font world trade GCM contents 251
- CICS services used 304
- control words
 - changes to 233
- current formatting environment
 - restoring 18
 - saving 18
- dictionaries
 - read-only 89
 - root word 89
 - searching 93
 - specifying language of 89
- font modification 249
- fonts
 - Gothic highlight text 255
 - Gothic special text 257
 - Gothic text 253
 - Serif highlight text 256
 - Serif special text 258
 - Serif text 254
- fonts provided with 247

SCRIPT/VS (continued)

- formatting environment 18
 - description of 17–19
 - for keeps and floats 17
 - for named areas 17
 - for running headings and footings 17
 - named 18
 - parameters that define 17
 - three parts of 17
- host command check 295
- in a batch environment 9
- invoking 13
- logical processing by 43
- options check 295
- overview of 9–13
- processing input lines
 - beginning with a blank 63
 - beginning with a tab 63
- processing of GML 74
- redefining formatting conventions 63
- root word dictionaries provided with 89
- source files provided with 112
- summary of functions 11–13
- tailoring 111
- unsupported control words 230
 - .BT control word 230
 - .CO control word 230
 - .CW control word 230
 - .DI control word 230
 - .EB control word 230
 - .EP control word 230
 - .ET control word 230
 - .EZ control word 230
 - .FI control word 230
 - .FM control word 230
 - .FS control word 230
 - .FT control word 230
 - .HE control word 230
 - .HM control word 230
 - .HN control word 230
 - .HS control word 230
 - .JU control word 230
 - .NB control word 230
 - .NC control word 230
 - .NJ control word 230
 - .OB control word 230
 - .OP control word 230
 - .OT control word 230
 - .PP control word 230
 - .PS control word 230
 - .RT control word 230
 - .SL control word 230
 - .SY control word 230
 - .TB control word 230
 - .TT control word 230
 - .ZZ control word 230
- using as a preprocessor 13, 210
- using as a subroutine 13, 209

SCRIPT/VS (continued)

- using as postprocessor 209
- using with other programs 207–212
- SEARCH option 232, 296
 - specifying macro libraries with 64
 - specifying symbol libraries with 64
- searching
 - dictionaries 87
 - words 87
- SEGLIB option 232
- segment
 - mapped page 140
- selecting masters 191
- self-study courses 325
- separation masters xiii
 - .SM [Separation Masters] 192
 - creating 189, 191
 - default master 195
 - defining document elements 192
 - multiple-color documents 189
 - multiple-part forms 189
 - numbering masters 196
 - producing 196
 - SEPMASR 191
 - SEPMASR keyword 189, 192
- SEPMASR keyword 189
 - with .DF [Define Font] 192
 - with .DR [Define Rule] 193
 - with .OI [Overlay Include] 194
 - with .PO [PostScript] 194
 - with .SD [Shade Definition] 194
 - with .SI [Segment Include] 195
- SEPMASR option 105, 191, 233
 - active values 28
 - improving system performance with 106
- Serif
 - highlight text 256
 - special text 258
 - text 254
- service routines 291, 296
- setting labels 45
- SGML TextWrite 323
- SGML Translator DCF edition 323
- SINGLE option 233
- SMFF (SCRIPT Mathematical Formula Formatter) 325
- SORCDD option 123, 233
 - changing
 - SCRIPT command options 123
- sort key 181
- sort sequences 178
 - defining correct sequence 149
 - DSMIXTT1 149
 - for indexes 147
 - index
 - modules 184
- sort values 182
 - assigning 182

- sorting
 - hexadecimal 182
 - index language 25
- source documents, managing 211
- source files provided with SCRIPT/VS
 - DSMCSDM 113
 - DSMCSPDB 116
 - DSMeUOT 119
 - DSMLPBCS 130
 - DSMLPLDT 131
 - DSMLPPDT 136
 - DSMTFFIB 140
 - DSMTRNLT 144
 - DSMTSMDM 115
- space unit notations
 - PostScript 279
- Spanish 180
 - prefixes 101
- SPELLCHK option 105, 233
 - description of 85
 - improving system performance with 106
 - using 85
- spelling verification 12
 - dictionaries 170
 - fallibility of 88
 - SCRIPT/VS support for 85–88
 - verification process 87
- SPF-II
 - using to edit macro library members 67
 - using utility function of 67
- SPF/CMS 66
 - using to edit macro library members 66
- STAIRS 294
- STAIRS/VS
 - Condensed Text Format (CTF) Records 209
 - definition of 207
 - input restrictions 208
 - paragraph number
 - description of 208
 - printing of 209
 - resetting of numbering counter 208
 - producing input for 207–209
 - restrictions in formatting input for 207
- STAIRS/VS (Storage and Information Retrieval System/Virtual Storage) 207
- stem processing
 - description of 93
 - prefixes removed during 93–101
 - suffixes removed during 93–101
- STOP option 233
- storage
 - minimizing 105
- Storage and Information Retrieval System/Virtual Storage (STAIRS/VS)
 - See STAIRS/VS
- Storage and Information Retrieval System/Virtual Storage (STAIRS) 207
- Structured Programming Facility-II
 - see SPF-II
- suffixes
 - Canadian French 95
 - Dutch 96
 - English 94
 - French National 95
 - Italian 97
- suffixes removed during stem processing 93
- Summary of Amendments xiii
- SVTEXT 215
- SYM parameter
 - of .LY control word 65
- symbol length attribute 49
- symbols
 - attributes 39
 - &a' 39
 - &AD' 39
 - &DH' 39
 - &DV' 39
 - &E' 39
 - &L' 39
 - &r' 39
 - &SD' 39
 - &SH' 39
 - &SV' 39
 - &SW' 39
 - &T' 39
 - &U' 39
 - &V' 39
 - &W' 39
 - &X' 39
 - beginning with an asterisk (*) 30
 - comparing null values 46
 - conditional processing with 43
 - containing special characters 47
 - control word separator 72
 - counter 61
 - defining in a macro library 65
 - defining within a macro 57
 - extended processing of 35
 - figure referencing 61
 - list of figures 61
 - logical processing with 49
 - numbering 61
 - processing 11
 - redefining 211
 - set by tokens 29
 - set when file is appended 29
 - set when file is imbedded 29
 - set when macro is processed 30
 - set with SCRIPT command 29
 - starting with control word separator 35
 - substituting values for within a macro 56, 57
 - symbol substitution with macros 56
 - using as associative memory 61
 - using for arrays of values 31

- SYOFF option 233
- SYON option 233
- SYSOUT comparand 46
- SYSPAGE comparand 46
- system performance
 - considerations 105
 - improving 106, 107
 - with the FPASSES(n) option 106
 - with the INDEX option 106
 - with the SEPMASTR option 106
 - with the SPELLCHK option 106
 - with the TWOPASS option 107
- system symbols
 - &\$ASCI 24
 - &\$DCF 24
 - &\$DDUT 24
 - &\$EGML 24
 - &\$ENV 25
 - &\$ENVARRAY 25
 - &\$GML 25
 - &\$IXLG 25
 - &\$LC 26
 - &\$LDEV 46
 - &\$LST 26
 - &\$PASS 26
 - &\$PDEV 46
 - &\$PRT 26
 - &\$PSNO 27
 - &\$RET 27
 - &\$RF 27
 - &\$RH 27
 - &\$SEPM 28, 191
 - &\$SVLG 28, 313, 314
 - &\$TAG 28
 - &\$TAGD 28
 - &\$TWO 27
 - &\$VR 28
 - for control values 23
- systems, office 13
- SYSVAR option 233
 - setting symbols with 29

T

- tables
 - changing in profile 202
 - changing outside table rule 202
 - punctuation 144
 - word delimiter characters 144
- tabs
 - processing input lines that begin with 63
- tag set for memos xiv
- tag set for overhead transparencies xiv
- tag set for schedules xiv
- tags
 - :GDOC
 - LANGUAGE attribute 169
 - attributes 72
 - name requirements 72

- techniques for logical processing
 - See logical processing by SCRIPT/VS
- TERM option 233
- text
 - editor
 - ATMS-III 9
 - editors
 - in a CMS environment 9
 - processing
 - using SCRIPT/VS for 9
 - residual 73
 - text processing, logical 43
 - text variables
 - changing fonts of 38
 - defining 36, 37
 - overriding delimiter characters 37
 - producing special characters 36
 - TextTagger ProcessMaster edition 323
 - TextTagger Workstation edition 323
 - TLIB option 233
 - description of 92
 - tokens
 - definition of 30
 - used in passing symbol values 30
 - tracing processing actions and debugging 13
 - training courses 325
 - TRANSLATE option 233
 - translation
 - character in a key to their header characters 183
 - single character to 1- or 2-character string 181
 - tables
 - building 147
 - uppercase string 173
 - TSO
 - changes xiv
 - creating macro libraries in 66
 - message definitions 115
 - using with SCRIPT/VS 9
 - TSO/FORMAT
 - creating a compatible environment with 242
 - unsupported control words 243
 - TWOPASS option 26, 27, 105, 233
 - improving system performance with 107
 - TYPE parameter 261

U

- UNFORMAT option 233
- unsupported control words
 - .BT control word 230
 - .CO control word 230
 - .CW control word 230
 - .DI control word 230
 - .EB control word 230
 - .EP control word 230
 - .ET control word 230
 - .EZ control word 230
 - .FI control word 230
 - .FM control word 230

unsupported control words (*continued*)

- .FS control word 230
- .FT control word 230
- .HE control word 230
- .HM control word 230
- .HN control word 230
- .HS control word 230
- .JU control word 230
- .NB control word 230
- .NC control word 230
- .NJ control word 230
- .OB control word 230
- .OP control word 230
- .OT control word 230
- .PP control word 230
- .PS control word 230
- .RT control word 230
- .SL control word 230
- .SY control word 230
- .TB control word 230
- .TT control word 230
- .ZZ control word 230

UPCASE option 127, 233

update transaction records

- definition of 157

- format of 160

uppercase string translation 173

user dictionaries

- building 91

- creating and maintaining 157

- definition of 89

- See also dictionary maintenance process

using to set current page number 31

words

- searching order 87

world trade fonts, creating 250

WRITE TERMINAL requests 300

V

variables

- defining 36

- local 59

- processing 59

verifying spelling 85

vertical rules

- currently drawn 28

VM and MVS extended architecture xiii

VSE

- running SCRIPT/VS under 9

W

WORD parameter

- of .DC control word 86

word-processing systems

- using with host systems 9

WORDHYBF 316

WORDHYCH 317

WORDHYLF 316

WORDHYLG 316

WORDHYWS 317

Readers' Comments

Document Composition Facility SCRIPT/VS Text Programmer's Guide

Publication No. SH35-0069-06

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

Note: IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

- | | Yes | No |
|---|--------------------------|--------------------------|
| • Does the publication meet your needs? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Did you find the information: | | |
| Accurate? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to read and understand? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to retrieve? | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use? | <input type="checkbox"/> | <input type="checkbox"/> |
| Legible? | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete? | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated? | <input type="checkbox"/> | <input type="checkbox"/> |
| Written for your technical level? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Do you use this publication: | | |
| As an introduction to the subject? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a reference manual? | <input type="checkbox"/> | <input type="checkbox"/> |
| As an instructor in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a student in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| • What is your occupation? | | |

Thank you for your input and cooperation.

Note: If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

Comments:[illegible]

Name _____

Address

Company or Organization

Phone No.

Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 588
P.O. Box 1900
Boulder, CO 80301-9191



Fold and Tape

Please do not staple

Fold and Tape

Readers' Comments

Document Composition Facility
SCRIPT/VS Text Programmer's Guide
Publication No. SH35-0069-06

Use this form to provide comments about this publication, its organization, or subject matter. Understand that IBM may use the information any way it believes appropriate, without incurring any obligation to you. Your comments will be sent to the author's department for the appropriate action. Comments may be written in your language.

Note: IBM publications are not stocked at the location to which this form is addressed. Direct requests for publications or for assistance in using your IBM system, to your IBM representative or local IBM branch office.

- | | Yes | No |
|---|--------------------------|--------------------------|
| • Does the publication meet your needs? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Did you find the information: | | |
| Accurate? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to read and understand? | <input type="checkbox"/> | <input type="checkbox"/> |
| Easy to retrieve? | <input type="checkbox"/> | <input type="checkbox"/> |
| Organized for convenient use? | <input type="checkbox"/> | <input type="checkbox"/> |
| Legible? | <input type="checkbox"/> | <input type="checkbox"/> |
| Complete? | <input type="checkbox"/> | <input type="checkbox"/> |
| Well illustrated? | <input type="checkbox"/> | <input type="checkbox"/> |
| Written for your technical level? | <input type="checkbox"/> | <input type="checkbox"/> |
| • Do you use this publication: | | |
| As an introduction to the subject? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a reference manual? | <input type="checkbox"/> | <input type="checkbox"/> |
| As an instructor in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| As a student in class? | <input type="checkbox"/> | <input type="checkbox"/> |
| • What is your occupation? | | |

Thank you for your input and cooperation.

Note: If mailed in the U.S.A., no postage stamp is necessary. For residents outside the U.S.A., your local IBM office or representative will forward your comments.

Comments:[illegible]

Name _____

Address

Company or Organization

Phone No. _____



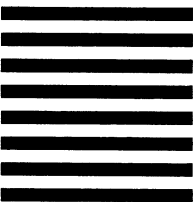
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 588
P.O. Box 1900
Boulder, CO 80301-9191



Fold and Tape

Please do not staple

Fold and Tape



File Number: S370-40
Program Number: 5748-XX9

SH35-0069-06

