

Computer Systems Division

HP Data Entry and Forms Management System (VPLUS/V)

Reference Manual



19447 Pruneridge Ave., Cupertino, CA 95014

Part No. 32209-90001
U1187

Printed in U.S.A. 7/86
Updated 11/87

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

Copyright © 1978,1980,1981,1982,1986,1987 by HEWLETT-PACKARD COMPANY

MANUAL UPDATE



MANUAL IDENTIFICATION

Title: HP Forms Management (VPLUS/V) Reference Manual

Part Number: 32209-90001

Edition Date: July 1986

UPDATE IDENTIFICATION

Update Number: 1

Update Date: November 1987

THE PURPOSE OF THIS MANUAL UPDATE

is to accumulate all the changes to the latest edition of the manual. Earlier updates to the latest edition which have not been incorporated are contained herein. This update package consists of all new and changed pages (backup pages are provided when necessary) plus this cover letter.

CHANGED PAGES

have the date of the update at the bottom of the page. Changes are marked with a vertical bar in the margin; when an update is incorporated in a subsequent reprinting of the manual, these bars are removed. "New" pages are those which were not present in the latest edition of the manual.

TO UPDATE YOUR MANUAL

replace existing pages in the latest edition of the manual with corresponding pages from this update package. Destroy all replaced pages. In addition, insert any new pages from this update.



HEWLETT-PACKARD COMPANY
19420 HOMESTEAD ROAD, CUPERTINO, CA 95014

32209-90001
U1187

PRINTING HISTORY

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition or a new update is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software code printed alongside the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

First Edition	Nov 1978.	32209A.00
Second Edition	Jan 1980.	32209A
Update #1.	Apr 1980.	32209A.01
Update #1 Incorporated	Aug 1980.	32209A.01
Third Edition.	Feb 1981.	32209B
Fourth Edition.	Nov 1982.	32209B.02
Update #1.	May 1983.	32209B.03
Update #2.	Sep 1984.	32209B.04
Update #2 Incorporated	Apr 1985.	32209B.04
Fifth Edition	Jul 1986.	32209B.04.17
Update #1.	Nov 1987.	32209B.05.00

LIST OF EFFECTIVE PAGES

The List of Effective Pages gives the date of the current edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. Changes are marked with a vertical bar in the margin. If an update is incorporated when an edition is reprinted, these bars are removed but the dates remain. No information is incorporated into a reprinting unless it appears as a prior update.

Effective Pages	Date	Effective Pages	Date
v to vii	Nov 1987	3-53 to 3-54	Nov 1987
xi to xxiv.	Nov 1987	4-47 to 4-48.	Nov 1987
1-1 to 1-4	Nov 1987	6-1 to 6-4	Nov 1987
1-13 to 1-14a	Nov 1987	6-9 to 6-12b.	Nov 1987
3-1 to 3-4	Nov 1987	6-17 to 6-20.	Nov 1987
3-7 to 3-8	Nov 1987	6-43 to 6-44.	Nov 1987
3-15 to 3-18.	Nov 1987	6-49 to 6-50.	Nov 1987
3-33 to 3-34.	Nov 1987	6-63 to 6-66.	Nov 1987
3-37 to 3-40.	Nov 1987	6-85 to 6-92.	Nov 1987
3-45 to 3-50.	Nov 1987	7-1 to 7-4	Nov 1987

LIST OF EFFECTIVE PAGES (continued)

Effective Pages	Date	Effective Pages	Date
7-7 to 7-12.	Nov 1987	H-1 to H-2.	Nov 1987
7-15 to 7-20.	Nov 1987	H-5 to H-6.	Nov 1987
B-15 to B-16b.	Nov 1987	Appendix J	Nov 1987
B-23 to B-24j.	Nov 1987	Index	Nov 1987
B-45 to B-46b.	Nov 1987		
B-55 to B-58.	Nov 1987		
B-69 to B-70.	Nov 1987		
E-5 to E-6	Nov 1987		
G-1 to G-2b.	Nov 1987		
G-15 to G-20.	Nov 1987		

PREFACE

This publication is the reference manual for VPLUS/V, a comprehensive software system that operates on the HP 3000 computer system to provide terminal data entry and control for most HP terminals and data capture devices. VPLUS/V provides the following capabilities:

- Forms design, including data editing and formatting, and form modification.
- Stand-alone data entry program.
- Reformatting of entered data.
- Programmatic interface to terminals, files, and forms.



VPLUS/V can be used for stand-alone data entry with no programming effort, or it can be used as a front end to transaction processing applications written in any of the supported programming languages (listed in Appendix A.)

VPLUS/V users may be terminal users (such as data entry clerks), forms designers with or without programming experience, or programmers in any of the supported languages. The manual is organized to start with simple operation of the stand-alone data entry program, and progress through the various levels of form design and reformatting, to the intrinsics used for programmatic interface with forms and terminals.

In order to use the VPLUS/V system effectively, you should know how to operate the HP terminals used at your facility. If you use the programming interface, you may also need the manual for the programming language in which you are coding. The manuals in the following list contain all the information you might need as a supplement to this manual:

Using VPLUS/V (32209-90004)
HP 3000 Guide for the New User (32033-90009)
MPE Commands Reference Manual (30000-90009)
MPE Intrinsics Reference Manual (30000-90010)
MPE Software Pocket Guide (30000-90049)
Using Files (30000-90102)
Native Language Support Reference Manual (32414-90001)

Also, refer to the reference manual for the programming language to be used with VPLUS/V (listed in Appendix A), and the reference manual for your terminal.

Section 1 INTRODUCING VPLUS/V

Source Data Entry	1-1
Transaction Processing	1-1
Features	1-2
Using The Manual	1-4
Forms Designer	1-4
Applications Programmer	1-4
Application User	1-5
Designing Forms	1-6
Levels of Form Design	1-6
Simple Collection	1-6
Simple Editing	1-8
Field Type	1-8
Data Type	1-9
Field Name	1-9
Field Enhancement	1-9
Initial Value	1-10
Full Field Edits	1-10
Advanced Form Processing	1-11
Advanced Processing Statements	1-12
Save Fields & Other Global Specifications	1-13
Application-ready Buffer	1-13
Benefits of the ARB	1-13a
Program Interface	1-14
Data Entry Program	1-16
Data Collection	1-16
Browse/Modify	1-17
Reformatting Data	1-18

Section 2 ENTERING DATA WITH ENTRY

Protected and Unprotected Fields	2-1
Detecting Errors	2-1
Modifying Data	2-1
Reformatting Data	2-1
Using the Terminal	2-2
Function Keys	2-4
Display Enhancements	2-4
Printing Forms and Data	2-4
Running Entry	2-5
Specifying ENTRY Files	2-5
Forms File	2-5
Batch File	2-5
Removing Deleted Records From a Batch	2-7
Expanding the Batch File	2-7
Block Mode	2-8

CONTENTS (continued)

Local Form Storage	2-8
Forms Sequence	2-9
Repeating Forms	2-9
Appended forms	2-9
Frozen Forms	2-9
Entering Data	2-11
Optional and Required Fields	2-12
Program Generated Data	2-13
Correcting Errors	2-14
Before ENTER	2-14
After ENTER	2-14
System and Logic Errors	2-14
Interrupting Data Entry	2-15
Terminating a Session	2-15
Unexpected Program Interruption	2-15
Modifying Data	2-16
Entering Browse/Modify Mode	2-16
Resuming Data Collection	2-16
Viewing the Data	2-17
Modifying the Data	2-17
Deleting Data	2-17

Section 3

INTRODUCTION TO FORMS DESIGN

Getting Started with FORMSPEC	3-2
Specify a Forms File	3-2
Choose an Option	3-3
FORMSPEC Function Keys	3-4
Menu Sequence	3-5
Creating an Application-ready Buffer (ARB)	3-7
Terminating FORMSPEC	3-7
Unexpected Program Interruption	3-7
Ease of Forms Design	3-8
Form Layout	3-8
Field Delimiters	3-9
Field Tag	3-11
Defining the Fields	3-11
Field Number	3-12
Field Length	3-12
Field Name	3-13
Display Enhancement	3-14
Field Type	3-14
Data Type	3-15
Initial Value	3-17b
Understanding Form Sequencing	3-18
The Repeat Option	3-18
The Next Form Option	3-19
Sample of Forms Design	3-22

CONTENTS (continued)

Forms File	3-25
Forms Modification	3-25
Forms File Size	3-25
Expand Forms File	3-27
Compiling the Forms File	3-27
Listing Forms	3-28
Form Families	3-30
Creating Child Forms	3-30
Form Family Example	3-31
Using Terminal Features	3-33
The Security Enhancement	3-33
Color Enhancement	3-33
Terminal Color Pairs	3-33
Color Field Enhancements	3-33
Color Error Enhancements	3-34
Color Window Enhancements	3-34
Color Locks	3-34
Forms File Menu	3-35
Main Menu	3-37
Form Menu	3-40
Form Function Key Labels Menu	3-43
Form Layout	3-44
Field Menu	3-45
Globals Menu	3-48
Global Function Key Labels Menu	3-51
Save Field Menu	3-52
Using FORMSPEC to Create an ARB	3-53a
Setting Up the Data Type Conversion Record	3-53c
Generating the ARB	3-53d
Types of Field	3-53e
Using and Defining ARBs	3-53j
ARB Trace Facility	3-53k
Terminal/Language Selection Menu	3-54
Data Capture Device Specifications Menu	3-56

Section 4

ADVANCED FORMS DESIGN

Levels of Advanced Design	4-2
Entering Processing Specifications	4-3
Special Cases	4-4
Correcting Existing Specifications	4-4
Statement Syntax	4-5
Comments	4-5
Continuing Lines	4-5
Custom Error Messages	4-5
Checking Data Entered	4-6
Operands	4-9

CONTENTS (continued)

Field	4-10
Constant	4-10
Arithmetic Expression	4-11
Index Retrieve Operand	4-12
Processing Specifications for Configuration	4-13
DEVICE	4-14
LIGHT	4-15
LOCALEDITS	4-16
Processing Specifications for Field Edits	4-20
Edit Statements	4-20
LENGTH CHECK	4-21
SINGLE VALUE COMPARISONS	4-23
Character Comparisons	4-23
Numeric Comparisons	4-24
Date Comparisons	4-24
Native Language Support	4-24
TABLE CHECKS	4-25
Native Language Support	4-26
RANGE CHECKS	4-27
Native Language Support	4-28
CHECK DIGIT	4-29
PATTERN MATCH	4-30
Special Pattern Characters	4-30
Transparency	4-31
Choice	4-31
Range	4-32
Grouping and Optional	4-32
Repetition	4-33
Operator Hierarchy	4-34
Processing Specifications for Advanced Processing	4-36
SET	4-37
Assigning a Value to the Current Field	4-38
Moving Data Between Fields	4-40
Automatic Formatting	4-41
Character Type	4-41
Numeric Type	4-41
Date Type	4-42
Default Formatting	4-44
STRIP	4-46
JUSTIFY	4-47
FILL	4-48
UPSHIFT	4-49
CHANGE	4-50
IF	4-52
FAIL	4-55
STOP	4-56
PHASES	4-57
Configuration Phase	4-58
Initialization Phase	4-58

CONTENTS (continued)

Field Edit Phase.	4-60
Finish Phase	4-60

Section 5 REFORMATTING SPECIFICATIONS

Files.	5-3
Reformat File.	5-3
Relation of Forms to Output Records.	5-5
Input Forms Sequence.	5-5
Output Record Definition	5-6
Field Specifications	5-6
Combining Data from Several Forms	5-7
Separating Data into Several Records	5-9
Reformatting Repeating Forms.	5-10
Separating Data from One Batch File into Several Output Files	5-12
Using REFSPEC	5-13
Terminating REFSPEC	5-13
Unexpected Program Interruption	5-13
REFSPEC Function Keys.	5-14
REFSPEC Menus.	5-15
Reformat File Menu	5-17
Forms File Menu.	5-18
Main Menu	5-19
Adding a Reformat	5-19
Compiling the Reformat File	5-19
Adding Global Specifications	5-20
Changing the Forms File Name.	5-20
Modifying Reformat Specifications	5-20
Changing an Input Forms Sequence	5-20
Changing an Output Record Definition.	5-21
Deleting a Reformat.	5-21
Listing a Reformat.	5-21
Resuming Design from Main Menu	5-21
Globals Menu.	5-24
Input Forms Menu.	5-27
Output Record Menu	5-29
Output Field Menu	5-34
Data Type	5-35
Field Formatting.	5-35
Strip.	5-38
Justify	5-39
Sign	5-40
Fill.	5-42
Check Digit	5-43

CONTENTS (continued)

Running Reformat	5-44
Concatenating Batch Files	5-45
Using a User-Defined Command	5-45
Streaming REFORMAT	5-46

Section 6

USING VPLUS/V INTRINSICS

Multipurpose	6-1
Multilanguage	6-1
Error Detection.	6-1
How Intrinsics are Used	6-4
Form Definition Area.	6-4
Data Buffer Area	6-4
Window Area	6-6
Key Label Area.	6-6
Calling VPLUS/V Intrinsics.	6-7
Parameter Types	6-7
Communication Area.	6-9
Error Handling.	6-22
Intrinsic Call or File Errors	6-22
Editing Errors.	6-22
Error Messages	6-22
Determining Fields in Error.	6-23
Using Terminal Features.	6-24
The Touch Feature	6-24
Local Form Storage	6-24
The HP 2626A vs. the HP 2624B/HP 2394A	6-24
Form Storage Directory.	6-25
Loading Forms	6-25
Form Families and Local Storage	6-26
Appending Forms and Local Storage.	6-26
Purging Forms From Local Storage.	6-27
Intrinsic Descriptions.	6-28
Dependency Between Intrinsics.	6-31
VCHANGEFIELD.	6-32
VCLOSEBATCH.	6-34
VCLOSEFORMF.	6-35
VCLOSETERM.	6-36
VERRMSG	6-37
VFIELDDEDITS.	6-39
VFINISHFORM	6-41
VGETBUFFER.	6-43
VGETFIELD	6-45

CONTENTS (continued)

VGETFIELDINFO	6-48
Passing INFOBUF Without Entering Keys	6-49
Table Wraparound	6-50
VGETFILEINFO	6-51
VGETFORMINFO	6-54
Passing INFOBUF Without Entering Keys	6-55
Table Wraparound	6-56
Error Determination	6-56
VGETKEYLABELS	6-58
VGETLANG	6-60
VGETNEXTFORM	6-62
VGETtype	6-64
VINITFORM	6-68
VLOADFORMS	6-70
VOPENBATCH	6-72
Existing File	6-73
New File	6-73
Record Format	6-73
Labels	6-74
Creating Your Own Batch File	6-75
VOPENFORMF	6-76
VOPENTERM	6-78
VPLACECURSOR	6-80
VPOSTBATCH	6-82
VPRINTFORM	6-84
VPRINTSCREEN	6-85a
Modes of Operation	6-85b
Normal Mode	6-85b
Documentation Mode	6-85c
Limitations	6-85d
VPUTBUFFER	6-86
VPUTFIELD	6-88
VPUTtype	6-90
VPUTWINDOW	6-92
VREADBATCH	6-94
VREADFIELDS	6-96
VSETERROR	6-99
VSETKEYLABEL	6-102
VSETKEYLABELS	6-104
VSETLANG	6-106
VSHOWFORM	6-108
Performance Optimization	6-109
Display and Local Form Storage	6-109
Controlling the Keyboard	6-110
Controlling Preload of Forms	6-110

CONTENTS (continued)

Controlling the Touch Feature	6-111
Controlling the Bell	6-111
VUNLOADFORM.	6-112
VWRITEBATCH.	6-114
Data Collection Mode	6-115
Modify Mode	6-115

Section 7

USING FORMSPEC IN BATCH MODE

Compiling Forms File in Batch Mode	7-1
Updating Forms in Batch Mode.	7-1
Listing Forms Files in Batch Mode.	7-1
Relating Forms in Batch Mode	7-1
Invoking FORMSPEC in Batch Mode	7-2
Batch Mode Commands	7-3
COMPILE	7-5
COPY.	7-6
DELETE.	7-7
EXIT	7-8
FIELD	7-9
FILE	7-10
FKLABELS	7-11
FORMS	7-12
LIST	7-15
RELATE.	7-16
RENUMBER	7-17
SELECTLANG.	7-18
SELECTTERM.	7-19
ARB Batch Mode Commands	7-19a
ADDARBFIELD.	7-19b
ARBTOSCREEN.	7-19c
DELARB	7-19d
DELARBFIELD	7-19e
GENARB	7-19f
MODARBFIELD.	7-19g
MOVEARBFIELD.	7-19h
RENAMEARBFIELD.	7-19i
SCREENTOARB.	7-19j
Samples of FORMSPEC in Batch Mode.	7-20
Sample Command File	7-20
Sample Job Stream.	7-20

CONTENTS (continued)

Section 8

NATIVE LANGUAGE SUPPORT

Language Attribute	8-1
Unlocalized (NATIVE-3000)	8-1
Language-Dependent	8-1
International	8-2
NLS CAPABILITIES	8-2
Setting the Native Language ID Numbers	8-2
Field Edits.	8-3
Date Handling	8-3
Numeric Data.	8-3
Native Language Characters	8-3
Entry and Language ID Number.	8-4

Appendix A

SAMPLE PROGRAMS

SPL	A-2
COBOL.	A-26
FORTRAN 77	A-38
BASIC	A-57
TRANSACTION	A-66
PASCAL.	A-75

Appendix B

VPLUS/V ERROR MESSAGES

Error Messages	B-2
--------------------------	-----

Appendix C

USASCII CHARACTER SET IN COLLATING SEQUENCE

Appendix D

CHECK DIGIT CALCULATION

Modulus 10	D-2
Modulus 11	D-3
Alphabetic Check Digits.	D-4

Appendix E

APPLICATION HINTS

Designing your Forms File.	E-2
Rules for the DL Area	E-6
Coding the Touch Feature.	E-7

CONTENTS (continued)

Appendix F STATE CODES

Appendix G TERMINAL INFORMATION

Supported Terminals and Features.	G-2
Termtypes	G-2
Modified Data Tag.	G-2
Extended Local Edits	G-2
Relabeling Function Keys	G-2
Security Display Enhancement	G-2
Local Form Storage Capabilities	G-3
X. 25 Capability	G-4
Color Enhancement	G-4
Data Capture Device Special Features	G-4
The HP 264X Terminals.	G-5
HP 2640B Terminal.	G-5
HP 2644A Terminal.	G-6
The HP 262X Terminals.	G-7
HP 2624A Terminal.	G-7
HP 2624B Terminal.	G-7
HP 2625A, HP 2628A Terminals.	G-7
HP 2626A Terminal.	G-8
HP 2627A Terminal.	G-8
The HP 239X and HP 150 Terminals.	G-9
HP 2392A Terminal.	G-9
HP 2393A Terminal.	G-9
HP 2394A Terminal.	G-9
HP 2397A Terminal.	G-9
HP 150 Terminal.	G-9
The Data Capture Devices.	G-10
Terminal Communication Area	G-11
Terminal Buffer Configuration	G-13
Recovering from Unexpected Program Interruption.	G-14
User Environment Control File.	G-15
Advanced Terminal I/O Procedures.	G-16
VTURNON/VTURNOFF	G-16
VTURNOFF	G-17
VTURNON	G-19

CONTENTS (continued)

Appendix H VERSION CONTROL

KSAM File Management	H-2
Purge Forms File	H-2
Rename Forms File	H-2
Convert KSAM File	H-3
Adjust Menu	H-4
HP32209	H-5
The HP32209B Utility	H-6

Appendix I BIT MAP CONVERSION

Appendix J HP PRECISION ARCHITECTURE

HPPA Intrinsics	J-2
Introduction	J-2
VGETIEEEREAL	J-3
VGETIEEELONG	J-5
VPUTIEEEREAL	J-7
VPUTIEEELONG	J-9
The HP32209S Utility	J-11

FIGURES AND TABLES

FIGURES

Figure 1-1.	Overview of VPLUS/V	1-3
Figure 1-2.	FORMSPEC Form Menu	1-7
Figure 1-3.	FORMSPEC Screen Design	1-8
Figure 1-4.	Sample Field Menu	1-9
Figure 1-5.	Advanced Processing Specifications	1-11
Figure 1-6.	Transfers Controlled by VPLUS/V Intrinsic	1-15
Figure 2-1.	HP Terminal Keyboard	2-2
Figure 2-2.	Example of Forms Sequence	2-10
Figure 2-3.	Bracketed Fields	2-11
Figure 2-4.	Fields Without Brackets	2-12
Figure 2-5.	Example of Data Entry	2-13
Figure 2-6.	Example of Data Modification	2-18
Figure 3-1.	Forms File Menu	3-2
Figure 3-2.	Main Menu	3-3
Figure 3-3.	Relation Between Menus & Function Keys	3-6
Figure 3-4.	Sequence of Menus for Form Design	3-8
Figure 3-5.	Example of Form Layout	3-9
Figure 3-6.	Example of Form Layouts	3-10
Figure 3-7.	Example of a Field Menu	3-11
Figure 3-8.	The Field Attributes	3-12
Figure 3-9.	Example of a Field Extending over Several Lines	3-13
Figure 3-10.	Form Menu	3-18
Figure 3-11.	Sample Forms File Layout	3-23
Figure 3-12.	Forms File Prototype	3-26
Figure 3-13.	Forms File Listing	3-28
Figure 3-14.	Form Family Relationship	3-30
Figure 3-15.	Parent Form	3-32
Figure 3-16.	Child Form	3-32
Figure 3-17.	Forms File Menu	3-35
Figure 3-18.	Main Menu	3-37
Figure 3-19.	Form Menu	3-40
Figure 3-20.	Form Function Key Labels Menu	3-43
Figure 3-21.	Form Layout Menu	3-44
Figure 3-22.	Field Menu	3-45
Figure 3-23.	Globals Menu	3-48
Figure 3-24.	Global Function Key Labels Menu	3-51
Figure 3-25.	Save Field Menu	3-52
Figure 3-25a.	Menu Sequence for ARB Feature	3-53b
Figure 3-25b.	Globals Menu	3-53c
Figure 3-25c.	Data Type Conversions Menu	3-53d
Figure 3-25d.	ARB Menu	3-53e
Figure 3-25e.	Restructure ARB Menu	3-53h
Figure 3-25f.	ARB Field Menu	3-53j
Figure 3-26.	Terminal/Language Selection Menu	3-54
Figure 3-27.	Data Capture Device Specification Menu	3-56
Figure 3-28.	Form Layout in Forms spec	3-59

FIGURES AND TABLES (continued)

Figure 3-29.	Displayed on Standard Character Set Mini-CRT Screen	3-59
Figure 3-30.	Displayed on 32-Character One Line Alpha Display Screen . . .	3-59
Figure 3-31.	Displayed on Large Character Mini-CRT Screen	3-60
Figure 4-1.	Field Menu with Processing Specifications	4-3
Figure 4-2.	Field Menu With Custom Error Messages	4-6
Figure 4-3.	Field Menu with a Set Statement	4-44
Figure 4-4.	Flow of Phase Execution for Entry	4-59
Figure 5-1.	Relation Among Files Used for Reformatting	5-2
Figure 5-2.	Prototype of Reformat File	5-4
Figure 5-3.	Combining Data From 3 Forms into 1 Output Record	5-7
Figure 5-4.	Reformat Data From 3 Forms into 2 Output Records	5-8
Figure 5-5.	Separating Data From 1 form into Several Records	5-9
Figure 5-6.	Reformatting Data From Repeating Forms	5-10
Figure 5-7.	Reformatting Data Based on Form Sequence	5-11
Figure 5-8.	Generating 2 Output Files From 1 Batch File	5-12
Figure 5-9.	Relation of REFSPEC Menus to Function Keys	5-16
Figure 5-10.	Reformat File Menu	5-17
Figure 5-11.	Forms File Menu	5-18
Figure 5-12.	Main Menu	5-19
Figure 5-13.	Reformat Listing	5-22
Figure 5-14.	Globals Menu	5-24
Figure 5-15.	Input Forms Menu	5-27
Figure 5-16.	Output Record Menu	5-29
Figure 5-17.	Sample Output Record Menu	5-30
Figure 5-18.	Output Field Menu	5-34
Figure 6-1.	Operation of VPLUS/V Intrinsic	6-5
Figure 6-2.	Intrinsic Flow	6-29
Figure 6-3.	Intrinsic Dependencies	6-31
Figure H-1.	The Adjust Menu	H-4

TABLES

Table 2-1.	Function Keys for Entry	2-3
Table 3-1.	FORMSPEC Key Labels	3-4
Table 3-2.	FORMSPEC Reserved Word List	3-13
Table 3-3.	Display Enhancement	3-14
Table 3-4.	How Each Numeric Data Type Interprets entered Values	3-16
Table 3-4a.	Recommended Data Type Conversions	3-17a
Table 3-4b.	Valid Screen Data Types	3-17b
Table 3-4c.	Valid Application Data Types	3-17c
Table 3-5.	Valid Dates	3-17
Table 3-6.	Invalid Dates	3-17
Table 3-7.	Default Color Pairs	3-34
Table 3-8.	Forms File Menu	3-36
Table 3-9.	Form/ARB Relationships	3-36

FIGURES AND TABLES (continued)

Table 4-1.	Summary of Processing Statements	4-7
Table 4-2.	Terminal Local Edits	4-17
Table 4-3.	Local Edits Corresponding to Field Type Selections in the Terminal Edit Checks Menu	4-18
Table 4-4.	Local Edits Corresponding to Attribute Selection in the Terminal Edit Checks Menu	4-18
Table 4-5.	Local Edits Corresponding to Format Mode Selection in the Terminal Configuration Menu	4-19
Table 4-6.	Pattern Match Operators	4-34
Table 4-7.	Automatic Formatting for Character Data	4-41
Table 4-8.	Automatic Formatting for Numeric Data	4-42
Table 4-9.	Automatic Formatting for Dates	4-43
Table 4-10.	Conversion During Data Movement	4-43
Table 4-11.	Variations on the IF Statement	4-54
Table 5-1.	Key Definitions	5-14
Table 5-2.	Standard Formatting by Data Type	5-36
Table 5-3.	Examples Using STRIP	5-39
Table 5-4.	Examples of Three Types of Justification	5-40
Table 5-5.	Correspondence Between Signed Digits and Characters	5-41
Table 5-6.	Example Using the SIGN Option	5-41
Table 5-7.	Example Using the FILL Option	5-42
Table 5-8.	Example Using the FILL LEADING and FILL ALL Option	5-42
Table 6-1.	Summary of VPLUS/V Intrinsics	6-2
Table 6-2.	Examples of Intrinsic Call Formats for each Language	6-7
Table 6-3.	Data Types Used for Various Languages	6-8
Table 6-4.	Outline of Communications Area Contents	6-9
Table 6-5.	Communication Area Contents for Data Capture Devices	6-11
Table 6-6.	Codes for the Window Line Enhancement	6-13
Table 6-7.	Intrinsics by Function Group	6-28
Table 6-8.	Specifications Buffer	6-33
Table 6-9.	Actions Used by Intrinsics	6-40
Table 6-10.	Field Information Buffer	6-49
Table 6-11.	File Information Buffer	6-52
Table 6-12.	Form Information Buffer	6-55
Table 6-13.	Numeric Type Conversion	6-65
Table 6-14.	Correlation of VGETtype with the Data Types for each Language	6-66
Table 7-1.	FORMSPEC in Batch Mode Commands	7-3
Table A-1.	Program Languages and References	A-1
Table G-1.	Terminals Supported by VPLUS/V	G-3

VPLUS/V is a comprehensive software system that implements and controls source data entry. It also provides an interface between a terminal and any transaction processing program. Refer to Figure 1-1 for an overview of the entire VPLUS/V system.

Source Data Entry

As a source data entry system, VPLUS/V provides easy forms design with data editing and validation built into the forms. It also provides a ready-to-use data entry program called ENTRY that you can use to enter data without any programming effort. ENTRY allows you to "browse" the entered data and modify it as it is entered.

Thus, source data entry through VPLUS/V can be done without programming by simply using ENTRY. If, however, you need additional or different capabilities, you can write your own application incorporating VPLUS/V intrinsics. You may want to refer to the sample applications provided in Appendix A for guidance.

Transaction Processing

As an interface to transaction processing applications, VPLUS/V provides a set of intrinsics that allows you to control forms and data on a terminal from an application program. These intrinsics are available to programs written in any of the supported programming languages listed in Appendix A.

Additionally, some languages (also listed in Appendix A) provide the programmer with a special interface with terminals and forms, as described in their respective reference manuals. With these languages, the programmer does not call the VPLUS/V intrinsics directly. Instead, the programmer specifies the statements appropriate to the special interface.

VPLUS/V also provides a batch reformatting capability. You can enter specifications to control how entered data is to be reformatted, and then run a program to actually reformat the data.

The VPLUS/V intrinsics and the reformatting capability, either singly or in combination, provide a "front end" to existing transaction processing applications. Thus, VPLUS/V allows you to concentrate on processing problems rather than on editing data or controlling the terminal.

Features

The following are the main features of the VPLUS/V system:

- A forms design program (FORMSPEC) that allows quick and easy forms design at a terminal using "menus".
- Batch mode management of forms files (through FORMSPEC) that allows a forms file to be updated, compiled, and listed without tying up a terminal.
- Powerful runtime data transformation capability (Application-ready Buffer, or ARB) that allows data to be entered on the screen in an order and format that make sense to the user, while the application receives the data converted and reordered to match the format of its storage record.
- Advanced forms design (through FORMSPEC) that provides for editing, formatting, movement, and computation of data when the form is executed, using the user's native language for alphabetic information and the local customs for numeric and date information.
- A ready-to-run data entry program (ENTRY) that provides immediate data entry and modification with no programming effort.
- A flexible data reformatting design program (REFSPEC) that specifies reformatting of entered data.
- A batch program (REFORMAT) that reformats the data according to the REFSPEC formatting specifications and writes it to a file for use by an application.
- A set of intrinsics that provides a powerful programming language interface to terminals, using FORMSPEC definitions, from applications written in any of the supported programming languages.

Figure 1-1 shows data entered at the terminal. The data is then read and manipulated by VPLUS/V intrinsics in an application such as ENTRY. This manipulation is done according to FORMSPEC forms and editing specifications, in conjunction with the standard HP 3000 capabilities. The data entered at the terminal is written to a data entry "batch" file where it is then available for use by an application. If desired, it can be reformatted by REFORMAT and written to an "output file".

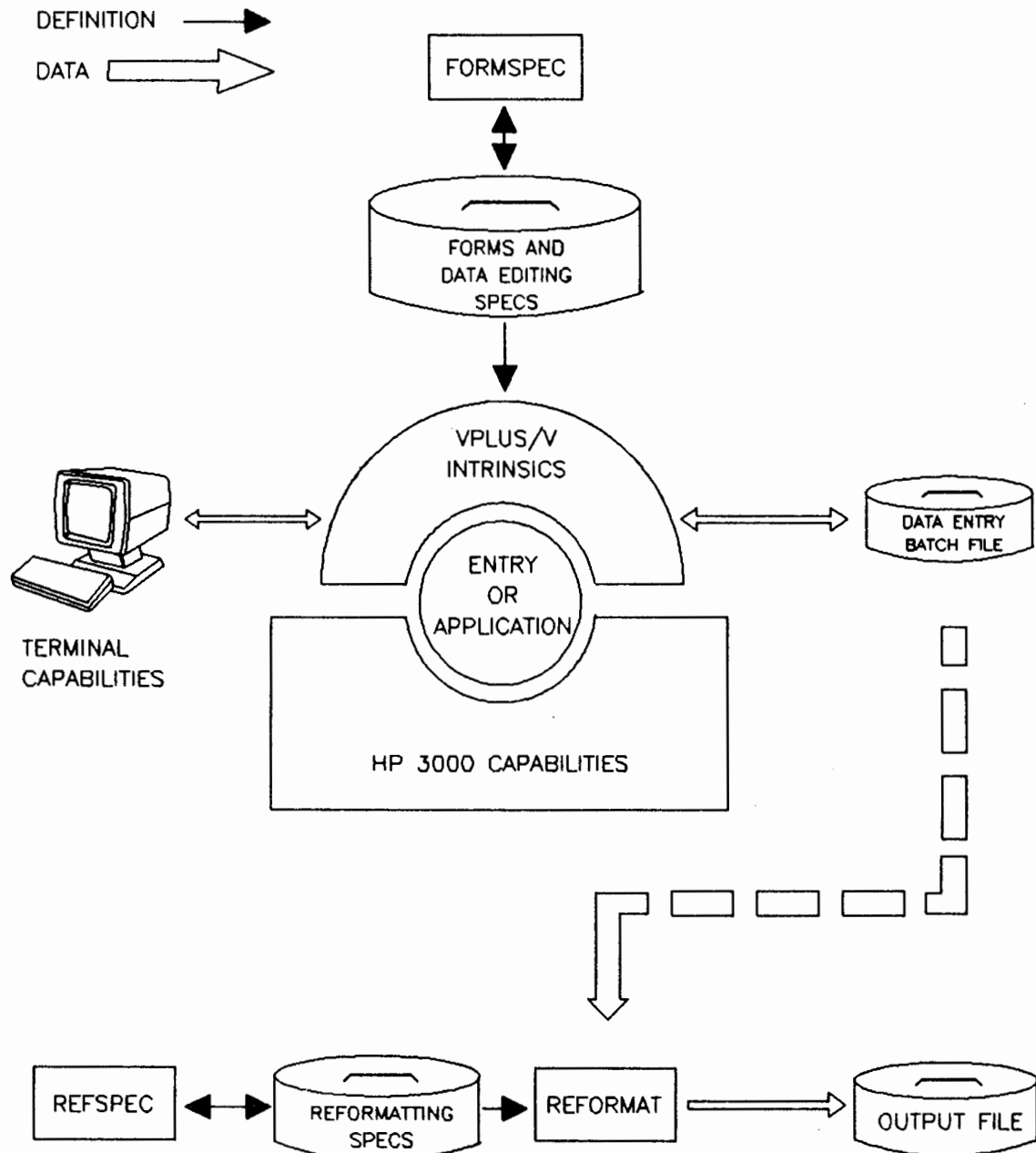


Figure 1-1. Overview of VPLUS/V

USING THE MANUAL

This manual may be useful to three different types of user:

- **Forms Designer** who designs the forms to be displayed, determines the order in which forms are displayed, and specifies any editing or special processing to be performed on data entered through the forms.
- **Applications Programmer** who designs and codes the application that uses data entered through the FORMSPEC forms, or displays data on these forms.
- **Application User** who enters data on the forms defined with FORMSPEC for processing by an application or by ENTRY.

Forms Designer

The forms designer who uses FORMSPEC to design forms should read Sections 3 and 4. These sections describe simple and advanced forms design with FORMSPEC. If designing forms in a native language other than the default (NATIVE-3000), the forms designer should refer to Section 8 for information on Native Language Support. The forms designer may also have the responsibility for reformatting the data entered through the forms. If so, the forms designer should also read Section 5, which describes how to specify reformatting with REFSPEC. Unless the forms designer is also the applications programmer, the forms designer can skip Sections 2, 6, and 7.

Applications Programmer

The first thing the applications programmer should determine is whether ENTRY, the data entry program provided with VPLUS/V, fulfills the user's application needs, either as is or with a few modifications. Using ENTRY from the user's point of view is described in Section 2.

If ENTRY is not suitable, the applications programmer should read Section 6 that describes the VPLUS/V intrinsics callable from any of the supported programming languages listed in Appendix A. If using one of the special interface programming languages, the applications programmer should refer to the reference manual for that language.

VPLUS/V Native Language Support enables an application programmer to create interactive applications which reflect native language numeric and date conventions in more than a dozen supported languages. For information on using VPLUS/V with Native Language Support, the applications programmer should refer to Section 8.

Since FORMSPEC provides editing and processing capabilities, the applications programmer should study FORMSPEC in order to determine how to divide the processing responsibilities between FORMSPEC and the application. Interactive FORMSPEC is described in Sections 3 and 4; batch mode FORMSPEC is described in Section 7.

The applications programmer should also read Section 5 that describes the specifications for reformatting data entered through an application or ENTRY. Reformatting may provide a way to adapt entered data to an existing application.

Save Fields & Other Global Specifications

Special fields, called "save fields" can be defined for the entire forms file. These fields are global to the forms file, and can be referenced in the processing specifications of any field in any form. Save fields are useful primarily in order to pass values between forms. They are defined on a special menu, the Save Field Menu.

Certain global characteristics of a form, such as default field or error enhancements, and the placement of the error/status line on the form are supplied by FORMSPEC as defaults. You may change these default characteristics through the Globals Menu.

Application-ready Buffer

An application-ready buffer (ARB) is a buffer passed between the application and the run-time intrinsics for data collection and display. It has these features:

- It stores data as the application will use it, which may differ from the way it appears on the screen.
- It is synchronized with other application sources, such as the program code and databases.
- It facilitates maintenance in that the user can rearrange fields on the screen without needing to modify the application program. Conversely, changes to the database affect the ARB, but not necessarily the screen.
- It supports an extended range of data types, including floating-point decimal and COBOL packed and zoned decimal.
- It simplifies coding because calls to VGETtype and VPUTtype are no longer needed: one call to VGET/PUTBUFFER is sufficient.

The application-ready buffer presents data as the application expects to use it, for example, to update a database, whereas the screen receives and displays data in the format that makes best sense to the user. The distinction between screen data and application-ready data facilitates maintenance and provides a mechanism for integrating FORMSPEC with a data dictionary if desired. If the definition of a database element changes, the ARB must be changed accordingly, but no change need necessarily be made to the screen. The designer can also change the screen layout, by altering the order of fields, for example, without having to modify the application source code.

The application receives the ARB from a VPLUS/V VGETBUFFER or sends it to the screen via VPUTBUFFER. The ARB returned to the application contains the data that the user entered, now transformed according to the designer's specifications. These may include adding, deleting and rearranging fields, as well as transforming and replacing data. A *mode* flag in the VPLUS *comarea* indicates whether the data is to be transformed or left "raw". As input to VPUTBUFFER, the ARB contains the information the application wishes to display on the user's screen.

Benefits of the ARB

The Application-Ready Buffer greatly simplifies screen/application data conversions. Previously, application programmers had to call VGET/PUTtype for each field that needed to be converted in a given transaction (or had to write code to perform conversions). Now they need only specify a default conversion for each screen and ARB data type, request automatic generation of the ARB from the form, and make any modifications that may be necessary for a particular field. This should considerably simplify the code, making it easier to read and maintain, and transaction response time should be unaffected.

The creation of an ARB is described in detail in Section 3.

PROGRAM INTERFACE

A library of high-level intrinsics is available for use by all terminal-oriented applications. These intrinsics can be called by any of the supported programming languages listed in Appendix A. Appendix A also contains listings of sample applications that use the intrinsics.

The program interface provides:

- | | |
|-------------------------|---|
| • Terminal Interface | Intrinsics to open and close a terminal file, to load and unload forms, to display a form to the terminal, and read data entered in fields of the displayed form. |
| • Forms File Interface | Intrinsics to open and close a forms file, to get the next form in sequence from the forms file, and to print the current form, with its contents, on the line printer. |
| • Data Manipulation | Intrinsics to initialize a form to its initial values, to perform any FORMSPEC or user-defined edits, and to perform any final form processing as defined by FORMSPEC. |
| • Access to Data | Intrinsics to read entered data to a program buffer, or write data from a program buffer; data from an entire form, or only from selected fields, can be read; similarly, data can be written for an entire form or only for selected fields. |
| • Data Transformation | Intrinsics to gather data entered by the user at the screen and transform it into the format most useful to the application, or vice versa. Data passed to or from fields can be converted to or from a variety of data types. |
| • Status/Error Control | Intrinsics to set or capture error flags and display messages. |
| • Batch Data Collection | Intrinsics that open and close the batch file, write data to the batch file, or read data from the batch file. |

The VPLUS/V intrinsics can be used for other purposes than data entry. Any application that wants to display forms on a terminal, accept data from or send data to a terminal, or validate entered data, can use the VPLUS/V intrinsics. The VPLUS/V intrinsics, as illustrated in Figure 1-6, can be used to:

- Transfer a form definition including processing specifications, from the forms file to the Form Definition area of memory.
- Display a form on the terminal screen.
- Transfer data from the form to the Data Buffer area of memory.
- Transfer data between the Form Definition and the Data Buffer areas of memory according to processing specifications.
- Transfer data between the Data Buffer in memory and either an application program buffer area or the batch data file used for data entry.

Thus, Figure 1-6 shows an overview of the various transfers between elements controlled by the VPLUS/V intrinsics. Note that many applications use their own files rather than the VPLUS/V batch file for the collected data.

INTRODUCTION TO FORMS DESIGN

SECTION

3

The forms displayed to a user are designed at a terminal with the interactive program, FORMSPEC. Using FORMSPEC, the designer designs the forms and defines parameters for editing, manipulating and displaying data. Forms defined in this manner are saved in a **forms file**. At any time during forms design, FORMSPEC allows the designer to modify the forms in the forms file. Once a form has been defined, the designer can create an application-ready buffer (ARB) for it. The ARB stores screen data in a format compatible with the requirements of the application program: this format may differ from the screen layout.

GETTING STARTED WITH FORMSPEC

You execute FORMSPEC in interactive mode by entering the following command in response to the MPE colon prompt:

```
:RUN FORMSPEC.PUB.SYS
```

FORMSPEC runs entirely in block mode. HP block mode terminals are placed in block mode automatically by VPLUS/V; refer to Appendix G for a list of supported terminals.

When you run FORMSPEC, it displays a series of "menus" on which you enter the specifications to define the forms for a single application. Each menu uses a fill-in-the-blanks type of format. Refer to the menu descriptions later in this section for more information on all the FORMSPEC menus.

With FORMSPEC, you press **ENTER** to record data; you use function keys to move between FORMSPEC menus and for other tasks, as described below. You can use your terminal editing keys to correct data typed into fields on the FORMSPEC menus; refer to your terminal manual for more information.

Most of the capabilities of interactive FORMSPEC are also available with batch mode FORMSPEC. (See "Using FORMSPEC in Batch Mode", Section 7 for more information.)

Specify a Forms File

The first menu displayed whenever you run FORMSPEC is the Forms File Menu, as shown in Figure 3-1. The letters v.uu.ff used for Figure 3-1, and for all the examples shown in this manual, will be replaced by the version number of VPLUS/V used on your system.

```
FORMSPEC v.uu.ff Forms File Menu

Forms File Name [FORMSF2.PUB.ACCTG]

PREV FORM  NEXT FORM  REFRESH  PREV  NEXT  MAIN/RESUME  EXIT
```

Figure 3-1. Forms File Menu

At the Forms File Menu, you specify the name of the forms file to be created or modified, such as the example FORMSF2.PUB.ACCTG shown in Figure 3-1; press **ENTER**. If the forms file does not exist, FORMSPEC prompts you to press **ENTER** again to confirm that you want to create a forms file.

Choose an Option

Once you have specified the name of the forms file and pressed **ENTER**, the next menu is the Main Menu, as shown in Figure 3-2. Notice that the top line, called the **window line**, contains useful information, which includes the version number (v.uu.ff), the title of the menu (Main Menu), and the name of the forms file (FORMSF2.PUB.ACCTG).

FORMSPEC v.uu.ff Main Menu		FORMS FILE: FORMF2.PUB.ACCTG
[A] Enter Selection Forms file is 1% full A--Add a form S--Add a Save field T--Terminal/Language Selection Menu G--Go to GLOBALS Menu, OR Go to form [] field [] L--List Forms File, OR List form ... [] D--Delete Save field [] Form [] C--Copy new form name [] from form [] from Forms File (opt) [] X--Compile Forms File Optional: Fast Forms File [] R--Relate child form [] to parent form [] B--Go to ARB Menu..... []		
PREV FORM	NEXT FORM	REFRESH
PREV NEXT MAIN/ RESUME EXIT		

Figure 3-2. Main Menu

From the Main Menu, you have access to the menus of FORMSPEC with all of their forms design features. You simply type the option desired in the selection box and press **ENTER**. In Figure 3-2, option **A** is specified, which selects "Add a form" and leads to a sequence of menus that allow you to design a form simply and easily. This process is described later under "Ease of Forms Design".

Many of the options, including **L**, **D**, **C**, and **X**, only require information in the fields on the Main Menu in order to be executed immediately. Option **B** lets you create and modify an application-ready buffer for a particular form.

Some options control the more advanced features of FORMSPEC, such as "Form Families" and "Using Terminal Features", discussed later in this section. All options are described under "Main Menu" in this section.

FORMSPEC Function Keys

In addition to the control provided by the Main Menu, the defined function keys, shown in Table 3-1, allow you to select menus for initial definition of an existing forms file, or for modifying it. The Main Menu and function keys together give you the ability to change any form, field, or global specification as you define the forms file.

Table 3-1. FORMSPEC Key Labels

Key	Key Action
PREV FORM F1	Display the previous Form Menu. If no forms are defined, a Form Menu with no values is displayed.
NEXT FORM F2	Display the next Form Menu. If the next form is not defined, a Form Menu with no values is displayed.
FIELD TOGGLE F3	Only used in Field Menus to switch between field attributes and the optional processing specifications.
REFRESH F4	Redisplay current menu in its initial state before any specifications were entered or existing specifications modified. Also used to recover in case of unexpected program interruption (refer to Appendix G).
PREV F5	Displays previous menu in sequence of menus (refer to "Menu Sequence").
NEXT F6	Displays next menu in sequence of menus (refer to "Menu Sequence").
MAIN/ RESUME F7	Request Main Menu or, if Main Menu displayed, return to menu displayed when MAIN/RESUME was requested.
EXIT F8	Terminate FORMSPEC and return to MPE control.

The function keys used by FORMSPEC should not be confused with the function keys defined by an application or by ENTRY for use while the application is executing, as discussed under "Advanced Forms Design" later in this section. Although the two groups of keys are physically the same, their functions differ in most cases.

Creating an Application-ready Buffer (ARB)

The purpose of the ARB is to allow you to transform screen data into application data and back again by converting data types between screen and application, and by reordering the fields.

Once you have created a form and saved it in the forms file, you can create an application-ready buffer (ARB) for it if you choose. Not every form needs to have an ARB; you can create ARBs for selected forms only. You can also create an ARB for a form at any time after you have created the form itself, and you can delete an existing ARB and generate a new one if you have made extensive changes to the associated form. Fields on the form are mapped one-to-one to the ARB when you select **GEN** from the ARB Menu. You can rearrange fields on the form using the Screen Design Menu without affecting the ARB, and you can add and delete ARB fields without affecting the form by selecting **RES** (Restructure) from the ARB Menu.

Before you can generate an ARB, you must define the Data Type Conversions you want to take place between the form on the screen and the ARB. From the Globals Menu, you access the Data Type Conversions menu and define the data conversion defaults. Screen and application data types are listed under "Data Type" further on in this section. Creation of an ARB is described in detail under "Using FORMSPEC to Create an ARB" in this section.

At runtime, the ARB transforms the data as it appears on the screen into a format that the application can use without further manipulation, and converts application data back into a format suitable for display on the screen.

Terminating FORMSPEC

You can terminate operation of FORMSPEC at any time by pressing **EXIT**. This returns you to MPE control which then issues the MPE colon prompt (:).

However, if you have made any modifications and have not compiled the forms file, FORMSPEC issues a warning. By pressing **EXIT** again you terminate operation of FORMSPEC without compiling the file. Refer to "Forms File" for more information on compiling the forms file.

Unexpected Program Interruption

In case of unexpected interruption due to hitting **(BREAK)** or a terminal power failure, control returns to MPE. Refer to Appendix G for the steps to recovery from such a situation. Once you have recovered, the menu will be cleared to initial or previously entered values. To ensure against damage to the file, reenter the information on all menus pertaining to the form you were creating or modifying at the time of the program interruption.

EASE OF FORMS DESIGN

Once you have specified a forms file and the Main Menu is displayed, you are ready to use FORMSPEC to create forms. Whenever you create a new forms file, your first task is always to add a form; if you are modifying an existing file, you may want to add a form. In either case, you follow the sequence of menus shown in Figure 3-4. Refer to the menu descriptions later in this section for more information on all the FORMSPEC menus.

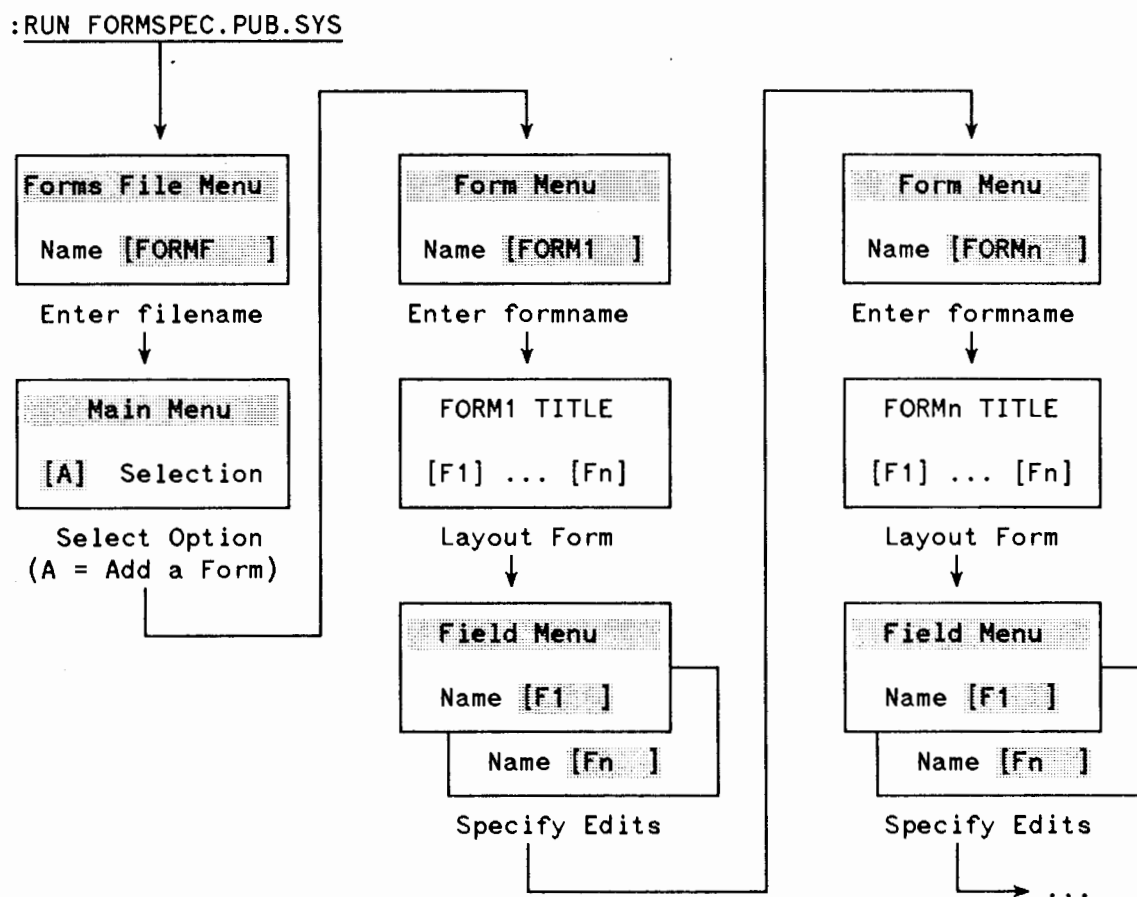


Figure 3-4. Sequence of Menus for Form Design

A Form Menu, shown later in Figure 3-10, allows you to define the characteristics of each form, such as the form name and what form sequencing options to use. Refer to "Understanding Form Sequencing" below.

Form Layout

The Form Menu is followed by a blank screen on which you design the layout of the form as it will appear on the screen. An example of a form layout is shown in Figure 3-5. This layout is easy to draw on the screen -- and easy to change at this stage or later. You use your terminal editing keys to layout the fields and text of your form on the screen; refer to your terminal manual for more information.

initial values may be displayed in any field, but such values can be, and usually will be, changed by the user.

Data Type

There are two sets of data types, the **screen** data types and the **ARB** data types. If you are creating an ARB for a form, the application needs to know the ARB data types. Each field on the form and the ARB must be specified as one of three data types: character, numeric, or date. Based on the data type, FORMSPEC can determine the basic validity of the data entered, how it is formatted for data movement, and the type of operations that can be performed on the field.

Screen data types impose format and edit rules for data entered on the screen. ARB data types determine how that data will be interpreted by the application. Conversions from screen data type to ARB data type and vice versa occur automatically at runtime (see VGET/PUTBUFFER in section 7).

Screen Data Type. Valid screen data types are: CHAR, NUM[*n*], DIG, IMP*n*, MDY, DMY, and YMD. Table 3-4 illustrates how each numeric data type interprets an entered value on the screen. The data types are described in the following paragraphs.

Default CHAR

Character Type.

CHAR Data entered in the field is assumed to be a string of any characters. No validity checking is performed on data at time of entry. No arithmetic operations can be performed on data of this type. The user can enter any characters in a CHAR type field. (As noted above, CHAR is the default data type.)

For example, \$12.59, A-15-75, and **123** are all legitimate entries.

Numeric Types.

NUM[*n*] Data entered in a field of this type must be numeric. The maximum number of decimal places can be indicated by the optional digit, *n*, where *n* is a value between 0 and 9. If you omit the value, *n*, the data item is assumed to have a floating decimal point.

Validity checking is performed on this data type. An optional leading sign (+ or -) is allowed. Commas are allowed, but if included they must be correctly placed. An optional decimal point may be included in the data. When Native Language Support is used, the symbols used to indicate thousands and decimals are language-dependent. For more information on Native Language Support, see Section 8.

The user must enter numeric data in this field. If *n* specifies a maximum number of decimal positions, the user must include a decimal point when the value has decimal places, and may omit the decimal point when it does not.

For example, in a NUM2 field, the user can enter a value with no decimal places (such as 500) or a value with one decimal place (such as 5.5) or a value with two decimal places (such as 5.95). Commas and a sign may also be included. For example, the following are legitimate entries for a NUM2 field: 1,390, -327.00,

+100,000.00 and so forth; but the value 5.678 is disallowed as it has too many decimal places.

DIG Data entered in a field of this type must be a positive integer. No sign, no commas, and no decimal point can be included. As with NUM data, validity checks are performed on this data type.

For example, 10030, 2 and 307 are all legitimate entries in a DIG field, but 10.5, 100,000 or -10 are not.

IMP n A value entered in an IMP field has an assumed decimal point. The assumed decimal point is n places from the right of the value. (Note that the value, n , must be specified.) Although permitted, a decimal point should not be entered in this type field, but commas and an optional leading sign (+ or -) are allowed. As with NUM type data, validity checks are performed on data of this type.

For example, if the data type is IMP2, and the user enters the value 500, the value is treated as 5.00.

With Native Language Support, the symbols accepted for thousands and accepted or required for decimal indication are language dependent. For more information on Native Language Support, see Section 8.

Table 3-4. How Each Numeric Data Type Interprets Entered

Values

Entered Value	Interpreted Value (based on data type)				
	NUM2	IMP2	NUM	NUM0	DIG
10.95	10.95	10.95	10.95	(error)	(error)
1095	1095.00	10.95	1095	1095	1095
10.9	10.90	10.90	10.9	(error)	(error)
10.956	(error)	(error)	10.956	(error)	(error)
-100	-100.00	-1.00	-100	-100	(error)
1,000	1,000.00	(error)	1,000	1,000	(error)
1,00000	(error)	1,000.00	(error)	(error)	(error)

Date Types.

MDY A date entered in an MDY field must be in the order: month, day, year. The data can be entered in any format, such as FEB 6, 1986 or 02/06/86 or FEBRUARY 6 86 and so forth.

- DMY** A date entered in a DMY field must be in the order: day, month, year. It can be any format, such as 2 MAR 1986 or 02-03-86 or 2/3/86 and so forth.
- YMD** A date entered in a YMD field must be in the order: year, month, day. It can be in any format such as 1986, APRIL 12 or 86/4/12 or 86-04-1 or 860402.

The entered data is checked by VPLUS/V for correct format and that it is a valid date. Arithmetic operations are not allowed on date fields.

Native Language Support does not affect the order of the date field. It does, however, accept the names of months and their abbreviations for each native language at run-time. For more information on Native Language Support, see Section 8.

To illustrate how the three date type specifications interpret entered dates for NATIVE-3000, Table 3-5 shows legal dates followed by Table 3-6 with illegal dates.

Table 3-5. Valid Dates

MDY	DMY	YMD
February 7, 1986	7 February 1986	1986, February 7
FEB 7 1986	7 FEB 1986	1986 FEB 7
02/07/86	07/02/86	86/02/07
2/7/86	7/2/86	86/2/7
02-7-86	07-2-86	86-2-07
2 7 86	7 2 86	86 2 7
020786	070286	860207

Table 3-6. Invalid Dates

MDY	DMY	YMD	Reason
February 7, 1986	7 February 1986	1986, February 7	Misspelled
FEBR 7 1986	7 FEBR 1986	1986 FEBR 7	Four letter abbreviation
2786	7286	8627	Must be a two-digit month, day when no separator is included.

ARB Data Types. The valid choices for ARB data types are: CHAR, INT, DINT, REAL, LONG, SPACK n , PACK n , SZONE n , ZONE n , and YYMMDD. Data type and length may differ from screen to ARB. The ARB data types include some that are language-specific. For example, SPACK n and PACK n correspond to COBOL COMP-3; SZONE n and ZONE n correspond to COBOL unsigned display numeric and signed display numeric respectively; and INT and DINT correspond to COBOL COMP. For more information on COBOL data types, see the *COBOL II/3000 Reference Manual*. The default data type and length for an ARB field are derived from the data type and length of the corresponding field on the associated form and the Data Type Conversion record.

Table 3-4a sets out recommended guidelines for data type conversions from screen to ARB and back.

Table 3-4a. Recommended Data Type Conversions

Screen Data Type --TO-->	ARB Data Type
CHAR	CHAR only
YMD, DMY, MDY	YYMMDD, CHAR
DIG	any except YYMMDD, CHAR
NUM[n], IMPn	any except YYMMDD, CHAR; "n" truncated if INT, DINT
ARB Data Type --TO-->	Screen Data Type
CHAR	CHAR
YYMMDD	MDY, YMD, DMY, CHAR, DIG
INT, DINT	DIG, NUM[n], IMPn; positive only if DIG
REAL, LONG, PACK, ZONE	DIG, NUM[n], IMPn; if DIG - "n" truncated, positive only

FORMSPEC will ensure the following length specifications for ARB data types:

YYMMDD length = 6
 INT length = 2
 DINT length = 4
 REAL length = 4
 LONG length = 8

YYMMDD is defined as a 6-byte ASCII field containing numeric data with no separators, in YMD order; for example, 860419.

The designer is responsible for making legitimate data conversions. The three critical factors are **data type**, **value**, and **length**. The following examples illustrate their importance:

- **Data Type:** Runtime errors may arise from specifying a CHAR/DATE source conversion to a numeric destination, or a CHAR source to a DATE destination.
- **Value:** The ARB data type INT may have a screen type of CHAR. This works if the field is for display only, but if it is an entry field, the user could input ABC unless the designer has taken steps to prevent it.
- **Length:** Data may be truncated if the screen data type DIG is specified for a field of length 10 and the ARB type is INT (length 2 bytes, equal to 1 HP 3000 word), since the largest number COBOL can store in one word is 4 digits and Pascal, 5. If the user enters 9 digits, a runtime error will result.

NOTE

These are recommendations only, and are not enforced by FORMSPEC edits. The programmer may set up the application code to handle non-standard conversions.

Table 3-4b shows the valid screen data types and their corresponding values for COBOL, Pascal and FORTRAN.

Table 3-4b. Valid Screen Data Types

ARB Data Type	COBOL	Pascal	FORTRAN
CHAR	PIC X (_)	Packed array CHAR [.._]	CHAR
YMD	as above	as above	as above
DMY	as above	as above	as above
DIG	as above	as above	as above
NUM[n]	as above	as above	as above
IMP[n]	as above	as above	as above

Note: [n] represents the number of decimal digits

Table 3-4c shows valid application data types and their values in COBOL, Pascal and FORTRAN.

Table 3-4c. Valid Application Data Types

FORMSPEC	COBOL	Pascal	FORTRAN
CHAR	X	Packed array CHAR [.._]	CHAR *_
YYMMDD	X(6)	as above	CHAR *6xx
ZONE _n	9(_). _n	---	---
PACK _n	9(_) _{Vn} COMP-3	---	---
SPACK _n	S9(-) _{Vn} COMP-3	---	---
REAL	---	REAL	REAL
LONG	---	LONG	Double Precision
INT	S9(4) COMP	Subrange -32768..32767	Integer **2
DINT	S9(9) COMP	Integer	Integer **4

Note: [n] represents the number of decimal digits

Initial Value

You may specify an initial value for the field. This value will be displayed in the field when the form is first displayed at the terminal. It is also displayed when **REFRESH** is pressed during data collection in ENTRY and the form is cleared to its initial values.

The value entered here is treated like user input. That is, it must match the data type of the field and must not be longer than the field length.

(Refer to Section 4 for a discussion of how FORMSPEC uses these data types to perform validity checks on data entered in the fields, and how data is treated during movement from one field to another.)

If Native Language Support is used, you must specify the initial value in NATIVE-3000. The value will appear with the conventions of the native language at run-time. For more information on Native Language Support, see Section 8.



Understanding Form Sequencing

As shown in Figure 3-4, the Form Menu is the first menu in the sequence of menus used to define a form. On the Form Menu, as shown in Figure 3-10, you specify a name for the form and the form sequencing options to use (Repeat Option and Next Form). The form sequencing options are described below. The other fields pertain to advanced features such as "Form Families" discussed later in this section.

FORMSPEC v.uu.ff Form Menu		FORMS FILE: FORMSF2.PUB.ACCTG	
Form Name	[FORM1]		
Repeat Option	[N] N--No Repeat A--Repeat, appending R--Repeat, overlaying		
Next Form	[C]	Name	[\$HEAD]
	C--Clear before Next Form A--Append Next Form F--Freeze, then append Next Form		
Function Key Label	[] Y--Define form level function key labels.		
Reproduced from	[] (opt)		
Comments	[]		
PREV FORM	NEXT FORM	REFRESH	PREV NEXT MAIN/ RESUME EXIT

Figure 3-10. Example of a Form Menu

The Repeat Option

When the forms in a forms file are displayed, a form may be repeated and appended to itself (A); it may be repeated overlaying the previous display of itself (R); or it can be a non-repeating form that is displayed once (N).

To illustrate how the three choices of the repeat option work, assume that form X is a repeat/append form (A). This form is displayed, the user types in data, and presses **ENTER**. The form with the data remains on the screen, and the same form with no data (except initial values) is displayed immediately below the form with data. The next time **ENTER** is pressed, the form is displayed a third time immediately below the second form. This continues until the repeat option is changed.

USING TERMINAL FEATURES

VPLUS/V takes advantage of special terminal features using processing specifications (refer to Section 4) or VPLUS/V intrinsics (refer to Section 6), with selections on the FORMSPEC menus. Two of these features, described below, are security and color. Refer to Appendix G for a list of terminals that support these features.

The Security Enhancement

To use the security enhancement, you should:

- choose the S type display enhancement on the Field Menu,
- select the appropriate terminal family on the Terminal/Language Selection Menu,
- use visible brackets only.

Refer to Appendix G for a list of terminals that support this enhancement.

Color Enhancement

VPLUS/V offers a simple way to provide color field, error and window enhancements for use with terminals that have the color feature. The addition of color to an application requires no programming changes; all color definitions are contained in the forms file.

A well-designed screen layout employing color improves the attractiveness and effectiveness of data presentation. The added dimension may simplify the expression of complex ideas and allow the representation of elusive concepts.

Terminal Color Pairs

There is a set of 8 default color pairs (see Table 3-7). VPLUS/V uses the terminal color pairs to implement color enhancements, with the terminal color pair 0 mapping to VPLUS/V color pair 8 and terminal color pairs 1-7 mapping to VPLUS/V color pairs 1-7. VPLUS/V assumes that an application has set its own values if the default values are not acceptable. It does not reinitialize the terminal's color pairs when the terminal is opened.

Color Field Enhancements

The Field Menu in FORMSPEC allows you to define color field enhancements by entering the digits 1-8 in the enhancement (ENH) field. Each field is enhanced at run-time in the color pairs initialized programmatically or by a terminal hard reset/power on.

VPLUS/V also allows the global definition of default field enhancements. The digits 1-8 are used at the Globals Menu to define a default color pair for all fields in a forms file.

Table 3-7. Default Color Pairs

VPLUS/V Color Pair	Foreground	Background
8	white	black
1	red	black
2	green	black
3	yellow	black
4	blue	black
5	magenta	black
6	cyan	black
7	black	yellow

The ENH fields in the Field and Globals Menus are four characters long, permitting a up to four different enhancements from the set of H, I, B, U, S or NONE. Thus, the combinations of color and display enhancements are limited. For example, a field defined with a combination of half bright, inverse video, underline and blinking (HIBU) cannot accommodate an extra character for color definition (HIBU2). If a color pair is specified with no other enhancements, NONE is assumed.

Color Error Enhancements

VPLUS/V supports global error enhancements by allowing the forms designer to define the desired enhancement on the Globals Menu at design time. This enhancement applies to all fields that appear in error at run time. Like the field enhancement, the error enhancement can be defined as up to four characters. One may be a digit from 1 to 8, indicating a color pair.

Color Window Enhancements

The window enhancement field on the Globals Menu in FORMSPEC allows you to define a four-character combination for enhancing the window. Like the field and error enhancements, a color pair, indicated by a digit, can be included.

Color Locks

FORMSPEC allows the forms designer to add color escape sequences to enhance the text of a form when it is being created. If the form is displayed on a HP 262X compatible terminal which does not support color, the color escape sequences are stripped. Therefore, the form displayed in FORMSPEC will no longer contain the color escape sequences when **ENTER** is pressed. To avoid the loss of valuable work, FORMSPEC will disallow color form modification on non-color terminals if an X is entered in the HP 2627A field on the Terminal/Language Selection Menu.

MAIN MENU

The Main Menu, as shown in Figure 3-18, is the main control menu for all FORMSPEC operations. If the forms file is new, you usually select **(A)** for Add a form. FORMSPEC then displays the menus that allow you to define your forms file, as described in "Ease of Forms Design". If the forms file already exists, you may enter any selection depending on what you want to do, such as add new forms to the file, select a particular menu in order to change it, delete a form, and so forth. Most of the options available on the Main Menu are also available using FORMSPEC in batch mode. (See Section 7 for more information.)

After modifying the forms file, you must compile it before it can be executed with the modifications. Only those forms of the existing file which have been changed are actually recompiled, unless a global attribute of the forms file, such as a save field, has been modified.

FORMSPEC v.uu.ff Main Menu		FORMS FILE: filename	
[] Enter Selection			
Forms file is 1% full			
A--Add a form			
S--Add a Save field			
T--Terminal/Language Selection Menu			
G--Go to GLOBALS Menu, OR Go to form	[]	field	[]
L--List Forms File, OR List form ...	[]		
D--Delete Save field	[]		
Form	[]		
C--Copy new form name	[]		
from form	[]		
from Forms File (opt)	[]		
X--Compile Forms File			
Optional: Fast Forms File	[]		
R--Relate child form	[]	to parent form	[]
B--Go to ARB Menu.....	[]		
PREV FORM	NEXT FORM	REFRESH	PREV NEXT MAIN/RESUME EXIT

Figure 3-18. Main Menu

Fields

A--Add a Form

To add a form, simply type **(A)** in the selection box and press **(ENTER)**. In response, FORMSPEC displays a Form Menu, as shown in Figure 3-19, so you can define the form. For details of form definition using this menu, see the description below under "Form Menu".

- S--Add a Save field** Specify **[S]** in the selection box and press **[ENTER]**. A blank Save Field Menu is displayed. This menu (see Figure 3-25) allows you to specify a save field description. To modify a save field, use **PREV** or **NEXT** to locate the particular Save Field Menu specification. For more information, refer to the "Save Fields Menu" description.
- T--Terminal/Language Selection Menu** Specify **[T]** in the selection box and press **[ENTER]**. The Terminal/Language Selection Menu is displayed. This menu (see Figure 3-26) allows you to specify the terminal or set of terminals as well as the native language (if utilizing Native Language Support) with which the forms file will be used. Refer to "Terminal/Language Selection Menu" below and Section 8 for detailed information.
- G--Go to Globals Menu** Type **[G]** in the selection box, but do not specify a form or field name. Press **[ENTER]**. The Globals Menu (see Figure 3-23) is displayed so you can make changes to the global characteristics of the forms file. It provides access to the Data Type Conversions Menu. (Refer to "Globals Menu" below for detailed information.)
- G--OR Go to form** To display the Form Menu for a specific form, type **[G]** in the selection box, specify the form name, and press **[ENTER]**. (You can also locate the Forms Menu with **PREV FORM** or **NEXT FORM**.) When the Form Menu is displayed, you can then change any specification on the menu. (Refer to "Form Menu" below for a discussion of the form description.)
- G--OR Go to field** The Field Menu for a specific field can be located by typing **[G]** in the selection box then specifying both the form and field name, and pressing **[ENTER]**.
- You can display or change field descriptions either on the Field Menu or directly during the form layout. The field attributes (except for length and number) and processing specifications can be entered and changed directly on the Field Menu. New fields are added, existing fields deleted, and the number and/or length of fields are changed by changing the form layout. (For details, refer to the "Ease of Forms Design" discussion earlier.)
- L--List Forms File OR List form** Type **[L]** in the selection box, specify the form name and press **[ENTER]** to print a description of any form in your forms file. If you want to list all the forms in the file with a description of the file in general, simply type **[L]** but do not specify a form name. Press **[ENTER]**. Refer to the example listing in Figure 3-13. The listings are printed on the standard list device (;DEV=LP, usually, the line printer) if there is no overriding MPE :FILE command. Consult Appendix E for information on the file equations and the JCW that VPLUS/V uses when listing forms files.
- D--Delete Save Field** To delete a save field, specify **[D]** in the Main Menu selection box, specify the save field name and press **[ENTER]**. The field will be deleted. You should be sure there are no references to the deleted field in the processing specifications of any other field (refer to Section 4).
- D--Delete Form** You can delete an entire form from the forms file by typing **[D]** in the selection box on the Main Menu, specifying the form name, and pressing **[ENTER]**. If the form has an ARB associated with it, the ARB is deleted at the same time.

**C--Copy new form
name from form
from Forms file (opt)**

To copy a form, type **(C)** in the selection box. Then specify the name you want to give the new form, the name of the existing form to be copied, and, if the form to be copied is not in the current file, the name of the file containing this form. Press **(ENTER)**. If the form has an ARB associated with it, the ARB will also be copied.

X--Compile Forms File

You can compile the current forms file by typing **(X)** in the selection box on the Main Menu and pressing **(ENTER)**.

Once the forms file has been compiled, only new and modified information is recompiled, unless a global attribute such as a save field, an item on the Terminal/Language Selection Menu, or an item on the Globals Menu has been modified. The newly compiled version replaces the previous version. There is never more than one compiled version of a forms file at a time. You may, however, modify the source version without affecting the compiled version. For example, if the compiled version is being used for data entry, it is not altered in any way as you modify the source version. When the source is completely modified, you can compile it and use the newly compiled version for data entry. In that case, the previous compiled version is lost.

**X--Compile Optional:
Fast Forms File**

To compile to a fast forms file, type **(X)** in the Main Menu selection box exactly as you would for any compilation (or recompilation), but also supply the fast forms file name and press **(ENTER)**.

**R--Relate child form
to parent form**

Form family relationships can be created by typing an **(R)** in the Main Menu selection box, and the names of the two forms which are to be "related" in the Relate fields and pressing **(ENTER)**. The forms must already exist and their form layouts must be identical. The forms must be distinct forms; that is, one form cannot be both the parent form and the child form. Furthermore, the child form cannot already be a child form or a parent form. Thus, it is also possible to relate forms which were never previously related. Family forms and their characteristics are fully discussed under "Form Families" earlier in this section.

B--Go to ARB Menu

To create, modify or delete an application-ready buffer (ARB), type in the Main Menu selection box, and the name of the associated form in the ARB Menu field, and press **(ENTER)**. The form must already exist in the forms file and the data type conversion record must have been set up using the Globals menu. Construction of an ARB is discussed in detail later in this section.

FORM MENU

The name and characteristics of each form are specified on the Form Menu. As soon as the Form Menu specifications are entered, FORMSPEC displays a blank screen on which you design the layout of the form. This form layout phase establishes the field names and their characteristics. Based on the form layout, FORMSPEC then displays a Field Menu for each field established on the form.

The Form Menu is displayed once for each form you define. When you have no more forms to define, press **MAIN/RESUME** to display the Main Menu in order to compile the forms file. The Form Menu is illustrated in Figure 3-19.

FORMSPEC stores forms in alphabetic order. When new forms are added to the forms file, they are inserted in the USASCII collating sequence. The Globals Menu (see description later in this section) allows you to specify which form is to be executed first. This form can be referenced by the system-defined value \$HEAD. If you do not specify the head form name, \$HEAD defaults to the first form in the file. Adding a new form may change the value of this default.

FORMSPEC v.uu.ff Form Menu		FORMS FILE: filename	
Form Name	[]		
Repeat Option	[R] N--No Repeat A--Repeat, appending R--Repeat, overlaying		
Next Form	[C] C--Clear before Next Form A--Append Next Form F--Freeze, then append Next Form	Name	[\$HEAD]
Function Key Label	[] Y--Define form level function key labels.		
Reproduced from	[] (opt)		
Comments	[]		
PREV FORM	NEXT FORM	REFRESH	PREV NEXT MAIN/ RESUME EXIT

Figure 3-19. Form Menu

You must specify at least a form name on the Form Menu. Default values are provided for the remaining specifications. If these defaults are not what you want, you can overwrite them with other values. If satisfied with the defaults, press **(ENTER)** to go directly to form layout.

FIELD MENU

If you want to assign special field characteristics, you can use a Field Menu (see Figure 3-22) for each field delimited during form design. Each menu displays up to three lines of the form, marks the field to which the menu applies, and displays the default attributes of that field. The Field Menus are presented in the order the fields appear on the form layout, from left to right and top to bottom.

You can change the default field attributes by typing new values and pressing **ENTER**. If you do not want to change the defaults, simply press **NEXT** to display the next Field Menu. If you are designing simple forms, in which all data is accepted and no special edits are performed, you can skip all the Field Menus by pressing **NEXT FORM**. If you change the field length or type, the screen length or type of the associated ARB (if any) is updated at the same time, but the ARB length and type are not changed.

The field attributes, all except length, can also be changed using FORMSPEC in batch mode. See Section 7 for more information. Processing specifications can be specified on the Field Menu. See Section 4 for a description.

```
FORMSPEC v.uu.ff Field Menu                                FORMS FILE: filename
```

Num	[n]	Len	[n]	Name	[fieldtag]	Enh	[HI]	FType	[0]	DType	[CHAR]
Initial Value	[]										

*** Processing Specifications ***

PREV
FORM

NEXT
FORM

FIELD
TOGGLE

REFRESH

PREV

NEXT

MAIN/
RESUME

EXIT

Figure 3-22. Field Menu

Fields

Num (Display only) A number between 1 and 256 assigned by FORMSPEC to the field at form layout. The number is permanently assigned to the field and is not changed even if other field attributes or the position of the field on the form layout changes. However, if the field tag is changed, a new field number is assigned to the field. The field numbers of a form can be renumbered in screen order with the FORMSPEC batch mode command RENUMBER, as described in Section 7.

Len (Display only) The length of the field as determined during form layout.

Name Specify any USASCII name of up to 15 characters long; it must start with an alphabetic character and may be followed by any alphanumeric (A-Z, a-z, 0-9) or the underline (_). It cannot be one of the reserved words listed earlier in Table 3-2.

Enh Specify any combination of enhancements using up to four characters:

H	Half Bright
I	Inverse video
U	Underline
B	Blinking
S	Security
1-8	Color (refer to Table 3-7)
NONE	No enhancement

Default HI

FType Specify the field type:

D	Display only -- cannot be modified by user.
R	Required -- cannot be left blank.
O	Optional -- may be left blank; edit checks and field phase processing specifications are skipped if the field is blank.
P	Process -- may be left blank, but edit checks are performed and field phase processing specifications are executed even if the field is blank.

Default O

DType

Specify the data type:

CHAR	Specifies data to be a string of any characters.
NUM[<i>n</i>]	Specifies data to be numeric. The maximum number of decimal places is indicated by the optional digit (<i>n</i>), where <i>n</i> is a value between 0 and 9. An optional leading sign (+ or -) is allowed; as are commas and decimal points if correctly placed.
DIG	Specifies data to be a positive integer. No sign, no comma, no decimal point allowed.
IMP <i>n</i>	Specifies data to be numeric with an assumed decimal point. The assumed number of decimal places is indicated by the required digit (<i>n</i>), where <i>n</i> is a value between 0 and 9. A sign and commas are allowed. A decimal point is allowed but should not be entered.
MDY	Specifies data to be a date in the order month, day, year.
DMY	Specifies data to be a date in the order day, month, year.
YMD	Specifies data to be a date in the order year, month, day.

When Native Language Support is used, the symbols used to indicate thousands and decimals are language-dependent. The names of months and their abbreviations are accepted for each native language at run-time. For more information on Native Language Support, see Section 8.

Default CHAR**Initial Value**

Specify data to be displayed in the field when the form is initially displayed.

GLOBALS MENU

Certain characteristics of the entire forms file are defined as global specifications and specified on the Globals Menu, illustrated in Figure 3-23. The global specifications include the default screen enhancements for fields and errors, the first form to be executed, where the message "window" line appears on the form, and provides for user-defined labels for the function keys. These specifications apply to the entire forms file rather than to individual forms or to fields within the forms. If you are planning to use the ARB feature, you must also supply global defaults for screen/ARB data conversions.

FORMSPEC supplies default values for these characteristics and, unless you want to change the defaults, you need never be concerned with global specifications. If, however, you want to change these specifications, you enter G in the Main Menu selection box, and do not specify a form name or a field name. The Globals Menu is then displayed so you can change the default global specifications. If you indicated the data capture devices on the Terminal/Language Selection Menu, the following message will appear at the bottom of the Globals Menu:

Press NEXT to select HP 3075/6 device specifications

(The Data Capture Device Specifications Menu is described later in this section).

FORMSPEC v.uu.ff Globals Menu		FORMS FILE: filename	
Head Form Name	[]		
Default Display Enhancement	[HI]		
Error Enhancement	[IU]		
Window Display Line	[24]		
Window Enhancement	[HI]		
Define Function Key Labels	[] ("Y")		
Define Data Conversions	[] ("Y")		
PREV FORM	NEXT FORM	REFRESH	PREV NEXT MAIN/RESUME EXIT

Figure 3-23. Globals Menu

When you press **ENTER**, FORMSPEC checks and records the global values (default or specified) and then displays the next menu. The next menu is the Save Field Menu. To get to the first Form Menu, press **NEXT FORM**.

Fields

Head Form Name

Enter the name of the first form you want displayed when the forms file is executed. If you leave this field blank, the first form in the forms file will be displayed. Adding forms to the forms file may change the first form. When the forms file is compiled, this name must identify an existing form. If you enter the name using lowercase letters, FORMSPEC shifts the letters to uppercase and then looks for a matching form name.

Default First form in USASCII collating sequence in the forms file. (\$HEAD)

Default Display Enhancement

When defining a form, you can specify individual display enhancements for any field on the Field Menu. If you do not specify them on the Field Menu, FORMSPEC assigns default field enhancements. If you do not want to use these defaults, you can specify your own default enhancements. Enter one or more of the display enhancement codes (B, H, I, S, U, and 1 - 8) or enter NONE if you do not want the fields to be enhanced at all. The enhancement codes can be entered in any combination, in any order. For example, for blinking, half bright, underlined, you can specify BHU, HUB, UBH. If you want to remove all display enhancements, enter NONE. The data capture devices do not support display enhancements.

Default Half bright, Inverse video (HI).

Error Enhancement

When a user entering data on a form makes an error, the field with the error is highlighted on the form by special display enhancements. You can change the default error enhancement by entering one or more of the display enhancement codes (B, H, I, S, U, and 1 - 8) in any combination, or you can enter NONE if you do not want fields with errors to be enhanced at all. For example, if you want fields with errors to be displayed in half bright, underlined, and to blink, enter BHU.

Default Inverse video, full bright, Underline (IU).

Window Display Line

You can specify the line of the terminal screen to be reserved for error and status messages. Screen lines are numbered from 1 (top) through 24 (bottom). You may enter 0 (zero) if you do not want a window. In this case, no error or status messages are displayed on the form during execution. Any fields in which errors are detected will still be enhanced when there is no window line, unless you specify NONE for error enhancements. If function key labeling on HP 264X terminals is in effect and a window line of 23 or 24 is chosen, line 22 is actually used.

Default Bottom line (24)

Window Enhancement

The window line is normally enhanced with inverse video, half bright. If you want a different enhancement for this line, you must specify it in this box. You may enter any of the standard enhancement codes (I, H, U, B, S, and 1 - 8), or you can specify no enhancements by entering NONE.

Default Inverse video, Half bright (HI).

Define Function Key Labels

To define function key labels, enter Y to obtain the Global Function Key Labels Menu. If you wish to retain the default global labels, leave this box blank. When function key labels have been defined, FORMSPEC will place an X in the Define Function Key Labels field. Once function key labels have been defined, the default labels can be reset by entering an N in this field. If you specify a Y in this field, the Global Function Key Labels Menu is displayed, as shown in Figure 3-24. The data capture devices do not support function key labeling.

Define Data Conversions

To use the ARB feature, you must specify default conversions for screen data to application data and vice versa. Type Y in this field to reach the Data Type Conversions Menu, shown in Figure 3-23a. The default data type conversions must be set up before you can create an ARB for a form.

To delete a save field, you also request the Main Menu and then enter D in the selection box and the name of the save field to be deleted in the field name box.

Fields

Save Field Name

A standard field identifier composed of up to 15 USASCII characters, beginning with an upper or lowercase letter. The remaining characters can be upper or lowercase letters, digits, or underlines. For example, SF3, D2_15, or ALPHA. Note that if you use a lowercase letter in a name, FORMSPEC shifts it to an uppercase letter. (The name must not be one of the reserved words listed earlier in Table 3-2.)

Length

Like other fields, save fields must be assigned a length, representing the maximum number of characters allowed in the field. Note that you must supply a field length, unlike field specifications where field length is determined from the screen design. Since save fields may be used as accumulators, it is important that the save field length be long enough to avoid rounding and/or truncation when summed values are moved to the field.

Data Type

The data types allowed are the same as for any field. The general types are character (CHAR), numeric (NUM[n], IMPn, or DIG), or date (MDY, YMD, or DMY). (Refer to "Ease of Forms Design" for a complete discussion of these data types.)

Initial Value

An initial value may be assigned to any save field. This is an optional specification; if omitted, all fields are set to blanks (\$EMPTY). If specified, the initial value must be the same type as the specified data type. Values are entered exactly as if they were entered by a terminal user; that is, characters are not delimited by quotes, and dates are not delimited by exclamation points. For example:

12.5	type is NUM1
3790	type is DIG
FEB 3, 86	type is MDY
John	type is CHAR

When the forms file is opened, initial values (default or specified) are assigned to the save fields.

USING FORMSPEC TO CREATE AN ARB

Once you have created a form and saved it in a forms file, you can create an associated application-ready buffer (ARB) for that form. There are two steps to the process; first, you must set up a data type conversion (DTC) record, then you can generate the ARB. The DTC record need only be set up once for the entire forms file.

You may want to transform data between the screen and application for several reasons. First, the data the application will store may differ from what appears on the screen: menu-selection and next-screen fields, for example, would normally be excluded from the ARB. Conversely, you may want to store data from a source other than the screen, such as key fields for an IMAGE dataset, along with the screen data. This can be done by using "filler" fields, fields that exist on the ARB but not on the corresponding form (see **RES** below).

Second, the order in which the application stores data may differ from the order in which it is entered on the screen. An arrangement of fields that makes logical sense to the user may not be suitable for a database, for example. Fields can be rearranged on the screen without affecting their order on the ARB, and vice versa.

Third, data type and length may differ from screen to ARB, and the designer uses the ARB screens, in conjunction with the Data Type Conversion Menu, to specify conversions (see discussion under "ARB Data Types").

Figure 3-25a shows the sequence of menus used to create, modify and delete an ARB and the fields on it.

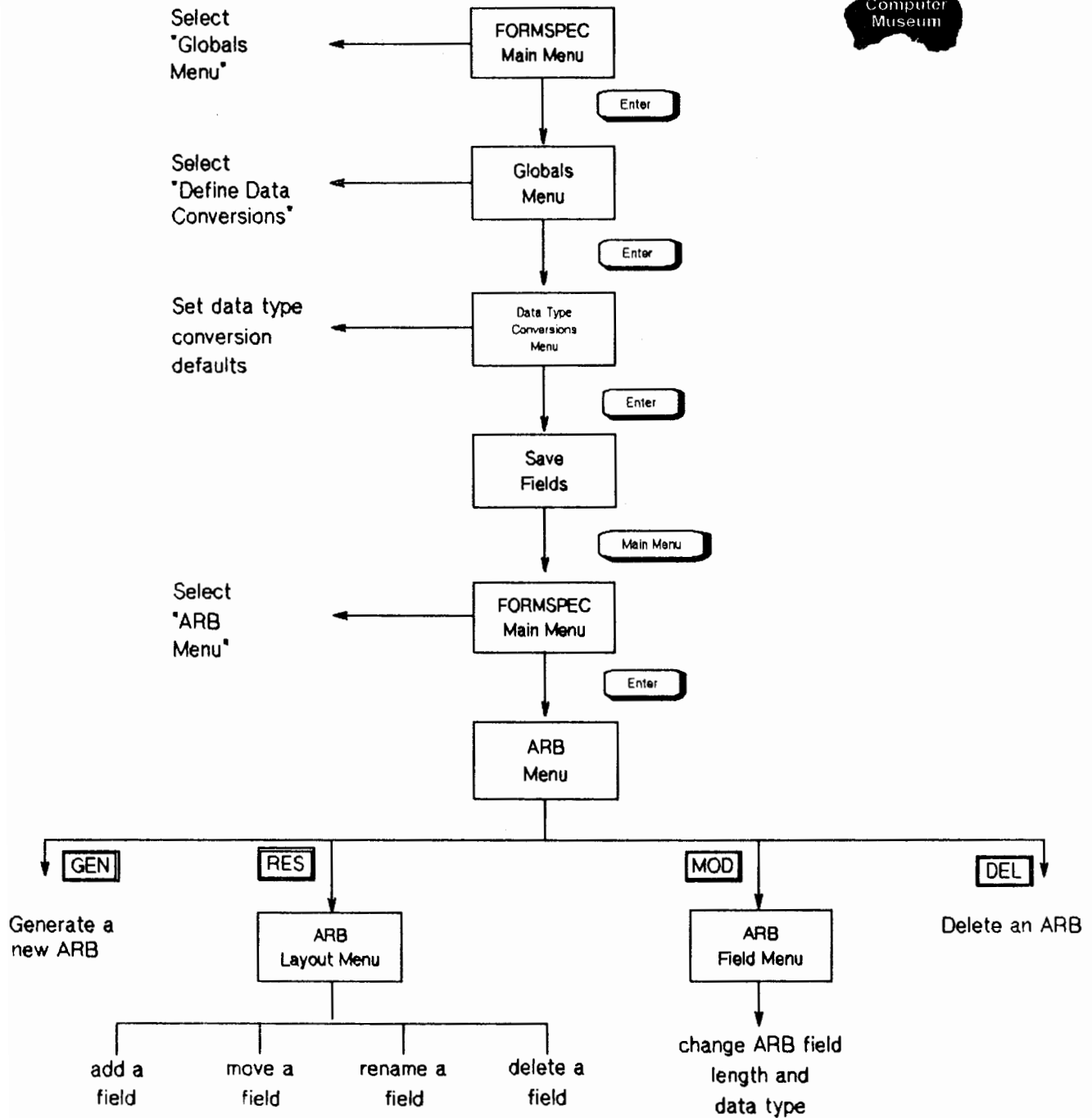


Figure 3-25a. Menu Sequence for ARB Feature

Setting Up the Data Type Conversion Record

The data type conversion record allows the forms designer to define the default values for screen and ARB data types. You **must** define these values, even if you only press **ENTER** to accept FORMSPEC's preset defaults, shown in Figure 3-25c. You will probably want to replace some, if not all, of these defaults with your own.

Set up the data type conversion record by following these steps:

1. Make sure you are at the FORMSPEC Main Menu and select **G** for "Go to Globals Menu". When you press **ENTER**, the Globals Menu is displayed.

FORMSPEC v.uu.ff Globals Menu		FORMS FILE: filename	
Head Form Name	[FORM 1]		
Default Display Enhancement	[HI]		
Error Enhancement	[IU]		
Window Display Line	[1]		
Window Enhancement	[NONE]		
Define Function Key Labels	[X] ("Y")		
Define Data Conversions	[Y] ("Y")		
PREV FORM	NEXT FORM	REFRESH	PREV NEXT MAIN/ RESUME EXIT

Figure 3-25b. Globals Menu

2. Enter **Y** for "Define Data Conversions" in the last box. The Data Type Conversions menu is displayed. This screen allows you to specify default conversions to and from every available screen and application data type.

FORMSPEC v.uu.ff Data Type Conversions Menu					FORMS FILE: FILENAME	
Default Data Type Conversions:						

From Screen Type to Application Type						
CHAR	date	DIG	NUMn	IMPn		
[CHAR]	[CHAR]	[CHAR]	[CHAR]	[CHAR]		

From Application Type to Screen Type						
CHAR	INT/ DINT	REAL/ LONG	SPACKn/ PACKn	ZONE _n	YYMMDD	
[CHAR]	[CHAR]	[CHAR]	[CHAR]	[CHAR]	[CHAR]	
PREV FORM NEXT FORM REFRESH PREV NEXT MAIN/ RESUME EXIT						

Figure 3-25c. Data Type Conversions Menu

The *n* in the numeric types stands for a number of decimal places. When specifying a conversion to one of these types, you must replace *n* with a digit or *a*. When you set the number of decimals to *a*, you are instructing FORMSPEC to determine the decimals algorithmically on the basis of the source type. Note that you can only select the *a* option on this menu, and not on the field menus. For recommendations on data type conversions, see "Data Types" earlier in this section.

3. When you have set your defaults, press **ENTER** and then press **MAIN/RESUME** at the SAVE FIELD Menu to return to the Main Menu.

Generating the ARB

Follow these steps to generate an ARB for a form.

1. At the Main Menu, enter **B** for "Go to ARB Menu" in the Main Menu Selection box.
2. In the "Go to ARB Menu" field, enter the name of the form for which you wish to generate an ARB and press **ENTER**. The ARB Menu is displayed.

FORMSPEC v.uu.ff ARB Menu	
Form Name [FORM1]	NO field(s) in ARB
[GEN] Enter selection	
GEN--Generate ARB	
RES--Restructure ARB (add, move and delete fields)	
MOD--Modify ARB field []	
DEL--Delete entire ARB	
REFRESH MAIN/ RESUME EXIT	

Figure 3-25d. ARB Menu

The form name is displayed in the first field. This is a display-only field: you cannot select a new form from this menu. If you want to create an ARB for a different form, you must go back to the Main Menu.

The next field shows the number of fields in the ARB: in this case there are none, since there is no ARB yet.

Types of Field

An ARB may contain two kinds of field. They are live fields and filler fields.

- Live fields exist on both the screen and the ARB.
- Filler fields exist on the ARB only; for example, alignment and accumulator fields.

The designer can select one of four options on the ARB Menu.

Options

GEN (Generate) Create an ARB from scratch, containing a field for each field on the form. The sequence of the fields accords with their sequence on the form, and can be altered using the Restructure ARB Menu (Figure 3-2c). The type and length

of the fields are determined by comparing the screen attributes with the data conversion record. They can be changed using the ARB Field Menu.

This option is only valid if no ARB exists for the form.

RES (Restructure)

Display the current ARB, showing the sequence of fields. Allows the designer to add, delete, move and rename fields using the Restructure ARB Menu.

MOD (Modify)

Modify the length and data type of the designated field in the ARB, using the Arb Field Menu.

DEL (Delete)

Delete the entire ARB. If you want to make a lot of changes to the fields on a form, it is usually better to delete the existing ARB and generate a new one once the form has been altered. If you copy or delete a form, the corresponding ARB will be copied or deleted as well.

Table 3-9 shows how an alteration made to the form will affect the ARB, if at all.

Table 3-9. Form/ARB Relationships

Form	ARB
Adding a form...	Does not add an associated ARB.
Copying a form...	Copies the associated ARB.
Renaming a form...	Renames the associated ARB.
Deleting a form...	Deletes the associated ARB.
Changing a form: - adding a field... - changing a field... - deleting a field...	Does not add it to the ARB once the ARB has been created, unless you create a new ARB. Changes the screen length and screen data type on the ARB, but does not affect the ARB field length and data type. Changes the corresponding ARB field to a "filler" field.
No effect on form...	Add, change, delete ARB or the fields in it.
Form can exist without an ARB.	ARB cannot exist without a form.

The Effect of Renaming a Field on a Form:

There are three possibilities, which are described below.

- **The old name is in the ARB but the new name is not:** The ARB field is renamed to the new name, retaining its "live" field type, and the screen length and screen type are altered if necessary.
- **The old name is not in the ARB, but the new name is, as a "filler" field:** The ARB field changes from "filler" to "live" field type, and the screen length and screen data type are updated if necessary.
- **Both old and new names exist in the ARB:** The *old* name changes from "live" to "filler" type, and the screen length and screen data type change to zero and blanks respectively. The *new* name changes from "filler" to "live" field type, and the screen length and screen data type are updated.

NOTE

FORMSPEC allows the designer to delete a field from a form after a corresponding field has been added to the ARB. The field need not be deleted from the ARB as well; at compile-time, the ARB field is converted to a "filler" field, with screen length set to zero and screen type blank. The designer can, however, choose to delete or rename the ARB field.

It is possible, though usually not advisable, to create an ARB without using the **GENERATE** feature. To do this, type **RES** and use the Restructure ARB Menu to add fields one at a time.

If your default conversion for any field(s) could cause run-time errors, you will get a message to that effect (see Appendix B, *VPLUS/V ERROR MESSAGES*). Go to the ARB Field Menu to check that the conversion does make sense in this case, or to change the target datatype if it doesn't.

3. Enter **GEN** in the Selection box and press **(ENTER)** to generate an ARB for this form. The "NO field(s) in ARB" changes to show a number equivalent to the number of fields on the form.
4. You can now change the position of a field in the ARB, add or delete fields, or modify a field by changing its length and/or data type.
5. To move, rename, add or delete fields, type **RES** in the Selection box and press **(ENTER)**. The Restructure ARB menu is displayed.

FORMSPEC v.uu.ff Restructure ARB Menu			
Form Name [FORM 1]		48 field(s) in ARB	
Enter selection: [MOVE] (Add, Move, Rename or DELETE)			
Field or Range: [FIELD A/X1]			
(Before, After): [AFTER] Destination: [ADDRESS2]			
001 FIELD A	002 FIELD D	003 FIELD C	004+XNOTFIELD
005 FIELD X	006 NAME	005+X1	006 ADDRESS
007+X2	008 CITY	009+HUNDREDS	010 THOUSANDS
011 ADDRESS2			
REFRESH		PREV PAGE	NEXT PAGE
		MAIN/RESUME	ARB MENU

Figure 3-25e. Restructure ARB Menu

This menu displays all the fields in the ARB by name and in sequence. The form name is displayed in the first field (display only). The lower part of the screen shows the ARB fields. Use **PREV PAGE** and **NEXT PAGE** to view them all if necessary.

- Use the fields at the top of the screen to change the order by moving fields or ranges. Press **ENTER** to execute all changes.

Options

Command

Must state MOVE, ADD, RENAME or DELETE. DELETE must be spelled out; the other commands may be abbreviated.

Field or Range

Name(s) or numbers(s) of the field(s). A slash (/) indicates a range: valid for MOVE or DELETE only. Any field may be deleted.

You can ADD only a single field at a time; it may not already exist on the ARB and it must conform to FORMSPEC naming rules. You may add a field that does not exist on the associated form. This "filler" field will have an ARB default length of 1 byte, and ARB type of CHAR; screen length is zero and screen type is blank. Fillers are marked with a "+" on the Restructure ARB Menu.

To **RENAME** a field, enter the name of an existing field, and enter the new name of the field in **DESTINATION**. "Live" fields exist on both form and ARB; "filler" fields exist on the ARB only. The following defaults apply to renamed ARB fields:

- **Live name changed to live name** (new ARB field has corresponding field on the form): new ARB field length and type derived algorithmically from the length and type of the corresponding field on the form.
 - **Filler name changed to live name** (new ARB field has corresponding field on the form): new ARB field length and type derived algorithmically from the length and type of the corresponding field on the form.
 - **Live name changed to filler name** (new ARB field has no corresponding field on the form): new field retains length and type of old ARB, and screen length becomes zero and type blank.
 - **Filler name changed to filler name** (new ARB field has no corresponding field on the form): new field retains length and type of old ARB, and screen length remains zero and type blank.
7. When you have finished modifying the field order on the ARB, press **ARB MENU** to return to the ARB Menu, or **MAIN/RESUME** to go back to the Main Menu.
8. To **change the length or data type** of an ARB field, type **MOD** on the ARB Menu. The ARB Field menu is displayed.

FORMSPEC v.uu.ff ARB Field Menu

Form Name [FORM1]

		---Screen---	-----ARB-----	
ARB Sequence #	Field	Len	Dtype	Byte Len Dtype
[1]	[A2]	[3]	[CHAR]	[3] [CHAR]

Figure 3-25f. ARB Field Menu

This screen allows you to scroll through all the fields in the ARB and change their length and data type if required.

NOTE

You can change only the **ARB** field type and length. To change the characteristics of the field on the **form**, you must use the FORMSPEC Field Menu (see Figure 3-22). If you use this menu to change the length or type of a field on the form, the corresponding screen length and screen type of the field in the ARB will be changed accordingly, but the ARB length and ARB type will remain the same.

Using and Defining ARBs

The programmer is responsible for:

- Using the correct data types for each programming language; for example, REAL is invalid for COBOL applications.

- Aligning and/or padding the ARB/source code. If an odd number of bytes is followed by an integer, some languages, including Pascal and FORTRAN, automatically pad the record definition, forcing the integer to be word-aligned. No such padding occurs in COBOL unless the SYNCHRONIZED clause is used (see the *COBOL II/3000 Reference Manual*).

If you use a language whose compiler pads to ensure word alignment for integers, you must pad your ARB correspondingly. For example, suppose you are coding in Pascal and you declare a record that looks like this:

```

ODD_BYTE_EXAMPLE = RECORD
  THREE_BYTES   : PACKED ARRAY [1..3] OF CHAR;
  GOTCHA        : INTEGER
  .
  .
  .
END;
```

The Pascal compiler will insert an additional byte after THREE_BYTES to align the integer on a word boundary. You must do the same with the ARB record; use the ARB LAYOUT screen to add a filler field after the three-byte field so that the ARB looks like this:

<i>Field Name</i>	<i>ARB Type</i>	<i>ARB Length</i>
DEPT	CHAR	3
FILLER1	CHAR	1
TOTAL PURCHASES	DINT	4

- Ensuring the application specifications match the ARB specifications; for example, if the ARB type is PACK, the COBOL specification should be COMP-3, not COMP.
- Avoiding run-time errors, which may occur when:
 - Converting a CHAR or date source to a numeric destination
 - Converting a numeric or CHAR source to a date destination
 - Invalid length specifications are encountered
 - There are alignment problems; FORMSPEC does not detect these

There is also the possibility that data will be truncated; for example, if DIG ---> INT, but *screenlen* is greater than 5.

The ARB Trace Facility

The ARB trace facility may be enabled by setting the JCW **VPLUSARBTRACE** to 1. This will print trace messages to the stdlist. You may direct VPLUS/V screens to a different device by using a **FILE** command to set the device of the filename used by **VOPENTERM** to a device other than the stdlist device.

This is an example of a trace from a form that has seven fields:

```
:SETJCW VPLUSARBTRACE,1
:FILE VTERM;DEV=99
:RUN ARBPROG
```

```
Field #1      : Buffer offset from 0 = 1
                Length = 2
                Type = INT
                Value = 1234

Field #2      : Buffer offset from 0 = 3
                Length = 2
                Type = INT
                Value = -1234

Field #3      : Buffer offset from 0 = 6
                Length = 2
                Type = INT
                Value = 1234

Field #4      : Buffer offset from 0 = 9
                Length = 4
                Type = DINT
                Value = 12345678

Field #5      : Buffer offset from 0 = 14
                Length = 4
                Type = DINT
                Value = -1234567

Field #6      : Buffer offset from 0 = 19
                Length = 4
                Type = DINT
                Value = 12345678

Field #7      : Buffer offset from 0 = 23
                Length = 8
                Type = CHAR
                Value = ABCDEFGH
```

```
END OF PROGRAM
:
```

The trace provides the following information. It shows:

- The location of each field in the ARB (Buffer offset);
- The length of each field in the ARB (Length);
- The type of transformation that occurs (Type);
- The value that is transformed in the ARB field (Value).

NOTE

This ARB is tied to seven form fields, but there are gaps in the buffer before field #1, and between each field from #2 to #6. These gaps are one-byte filler fields.

TERMINAL/LANGUAGE SELECTION MENU

The Terminal/Language Selection Menu allows you to specify which terminals will be used with the forms file. You may select one family or any combination of families, including all four families (HP 264X, HP 262X, HP 239X (includes HP 150) and HP 307X). The HP 264X family is the terminal default value. When the selection is defaulted, the forms file will run on the HP 264X family, the HP 262X/HP 239X family and the HP 150, but the following features of selected terminals will not be supported: color, local edits, function key labeling on HP 264X terminals, and security display enhancement. Consult Appendix G to see which HP terminals support which features. The family of terminals need not be specified unless the forms file will be making use of these terminal features or will be used with the data capture devices. Refer to the field descriptions below for more information.

The Terminal/Language Selection Menu also allows you to specify the native language with which the forms file will be used. See "Setting the Language ID Numbers", Section 8, for more information.

The options available on the Terminal/Language Selection Menu are also available using FORMSPEC in batch mode. (See Section 7, for more information.)

FORMSPEC v.uu.ff Terminal/Language Selection Menu		FORMS FILE: filename
Select (X) the terminals on which you will use this forms file: ☒ HP264X Family ☐ HP3075A, HP3076A, HP3081A ☐ HP262X, HP239X Families ☐ HP2627A, HP2397A ☐ Reserved ☐ Reserved		
----- Enter Language ID:		
	ID	Name
FORMSPEC Language	[0]	NATIVE-3000
Forms File Language	[0]	NATIVE-3000
PREV FORM NEXT FORM REFRESH PREV NEXT MAIN/ RESUME EXIT		

Figure 3-26. Terminal/Language Selection Menu

JUSTIFY

Used to move data within a field to the left or right boundary of the field, or center it within the field.

Syntax

JUSTIFY	{ LEFT RIGHT CENTER }
---------	-----------------------------------

Parameters

LEFT	The data in the current field is shifted to the left.
RIGHT	The data in the current field is shifted to the right.
CENTER	The data in the current field is centered in the field.

Discussion

The JUSTIFY statement can be used to format data entered into or moved to any field. (For automatic formatting, refer to the SET command earlier in this section.) As is true of all formatting statements, JUSTIFY has an immediate effect on the data in the memory buffer; the formatted data is then available for display or copying into the application. When the user enters data into a field for which formatting is specified, the data is formatted as soon as **ENTER** or an appropriate function key is pressed. Thus, FORMSPEC formatting is done before data is written to the data or batch file. (In contrast, reformatting through REFORMAT, described in Section 5, is done after data is collected, edited, and written to the batch file.)

Example

Statement	Data Entered	After Formatting
JUSTIFY LEFT	[ΔΔSMITHΔΔΔΔ]	[SMITHΔΔΔΔΔΔ]
JUSTIFY RIGHT	[ΔΔSMITHΔΔΔΔ]	[ΔΔΔΔΔΔSMITH]
JUSTIFY CENTER	[ΔΔSMITHΔΔΔΔ]	[ΔΔΔSMITHΔΔΔ]

If the data cannot be centered exactly, the extra blank is on the right.

FILL

Used to replace all the blanks between the field boundaries and the first or last nonblank data character with a designated character.

Syntax

FILL	{ TRAILING LEADING }	" <i>character</i> "
------	-------------------------	----------------------

Parameters

TRAILING	FILL TRAILING replaces those blanks following the data with the specified character.
LEADING	FILL LEADING replaces the blanks preceding the data with the specified character.
<i>character</i>	The specified <i>character</i> is any single character.

Discussion

The FILL statement can be used to format data entered into or moved to any field. (For automatic formatting, refer to the SET command earlier in this section.) As is true of all formatting statements, FILL has an immediate effect on the data in the memory buffer; the formatted data is then available for display or copying into the application. When the user enters data into a field for which formatting is specified, the data is formatted as soon as **ENTER** or an appropriate function key is pressed. Thus, FORMSPEC formatting is done before data is written to the data or batch file. (In contrast, reformatting through REFORMAT, described in Section 5, is done after data is collected, edited, and written to the batch file.)

Note that no formatting occurs when the field is blank.

Example

Statement	Data Entered	After Formatting
FILL TRAILING "*"	[Δ250ΔΔΔ]	[Δ250***]
JUSTIFY RIGHT FILL LEADING "0"	[Δ250ΔΔΔ]	[0000250]

Note that more than one formatting statement can be specified. Since the statements are executed in order of appearance, the FILL statement in the second example above affects the data justified by the preceding JUSTIFY statement.

A set of callable intrinsics is provided by VPLUS/V. Table 6-1 lists the VPLUS/V intrinsics in alphabetic order and summarizes their functions. A full description of each intrinsic appears later in this section. They can be used by any application, either for data entry or for other terminal related applications. The VPLUS/V intrinsics manage the interface between an application, the terminal, a forms file, the entered data, and, optionally, the batch file to which entered data may be written. These intrinsics are used by the VPLUS/V Data Entry Program (ENTRY) to control data entry.

Appendix A contains listings in different programming languages of sample applications. These listings provide useful examples of how to use the VPLUS/V intrinsics.



Multipurpose

The VPLUS/V intrinsics can be called by any application that displays forms on a terminal supported by VPLUS/V. With the exception of the batch file management intrinsics used specifically for data entry, the VPLUS/V intrinsics can be used in conjunction with FORMSPEC to allow for data base inquiries and updates via the application, as well as any other operation that results in the display or collection of data through a terminal form. Such applications can also make use of the capabilities that VPLUS/V provides for validating input data.

Multilanguage

The VPLUS/V intrinsics can be called from programs written in any of the supported programming languages listed in Appendix A. Additionally, some languages (also listed in Appendix A) provide the programmer with a special interface with terminals and forms, as described in their respective reference manuals. With these languages, the programmer does not call the VPLUS/V intrinsics directly. Instead, the programmer specifies the statements appropriate to the special interface. Each of the supported programming languages calls the VPLUS/V intrinsics with the same parameters, and these parameters are essentially the same type and size. So that the intrinsics can adjust to any peculiarities of the calling programming language, one input sub-parameter specifies the language of the calling program.

Error Detection

If a system or program error causes a VPLUS/V intrinsic to fail, an error code is returned to the calling program. Once an error has been detected, subsequent intrinsic calls do not perform any functions until the program detects the error and performs its own error routine. In order for processing to continue after an error is detected, the calling program must reset the error argument. This method of error handling means that intrinsic errors do not cause unexpected program termination.

Table 6-1. Summary of VPLUS/V Intrinsics

INTRINSIC	FUNCTION
VCHANGEFIELD	Changes field attributes for specified fields at run-time.
VCLOSEBATCH	Closes batch file.
VCLOSEFORMF	Closes forms file.
VCLOSETERM	Closes terminal file.
VERRMSG	Returns message associated with error code.
VFIELDEDITS	Performs field phase processing specifications.
VFINISHFORM	Performs final phase processing specified for form.
VGETBUFFER	Copies contents of data buffer into application.
VGETFIELD	Copies field contents from data buffer into application.
VGETFIELDINFO	Returns field information.
VGETFILEINFO	Returns forms file information.
VGETFORMINFO	Returns form information.
VGETKEYLABELS	Returns global or form function key labels.
VGETLANG	Returns the native language ID of the forms file being executed.
VGETNEXTFORM	Reads next form into form definition area of memory; window and data buffer are not affected.
VGETtype	Copies field contents from data buffer to application, converting data to specified type.
VINITFORM	Sets data buffer to initial values for form.
VLOADFORMS	Loads the specified forms into terminal local form storage memory.
VOPENBATCH	Opens batch file for processing.
VOPENFORMF	Opens forms file for processing.
VOPENTERM	Opens terminal file for processing.
VPLACECURSOR	Positions the cursor at a specified field after a form is displayed.

Table 6-1. Summary of VPLUS/V Intrinsics, continued

INTRINSIC	FUNCTION
VPOSTBATCH	Updates end of file mark in batch file after last record referenced.
VPRINTFORM	Prints current form and data buffer on offline list device.
VPRINTSCREEN	Prints entire contents of screen on offline list device.
VPUTBUFFER	Copies data from application to data buffer.
VPUTFIELD	Copies data from application to field in data buffer.
VPUTtype	Copies data of specified type from application to data buffer, converting data to external format.
VPUTWINDOW	Copies message from application to window area in memory for later display.
VREADBATCH	Reads record from batch file into data buffer.
VREADFIELDS	Reads input from terminal into data buffer.
VSETERROR	Sets error flag for data field in error and copies error message to window area.
VSETKEYLABEL	Temporarily sets a new label for a function key.
VSETKEYLABELS	Temporarily sets new labels for function keys.
VSETLANG	Specifies the native language ID to be used with an international forms file.
VSHOWFORM	Updates terminal screen, merging the current form, any data in buffer, any key labels, and any message in window.
VUNLOADFORM	Unloads a specified form from terminal local form storage memory.
VWRITEBATCH	Writes data from data buffer to batch file.

HOW INTRINSICS ARE USED

The VPLUS/V intrinsic control the interface between forms stored in a forms file, a terminal screen, data entered on a form, and an application. Figure 6-1 illustrates the relation between the VPLUS/V intrinsic, terminal, forms file, application, data file or optional batch file, and the buffer areas in memory used by the intrinsic.

The intrinsic use a buffer area in memory for the form definition and another buffer area for the data. A third area in memory is used as a buffer for the window area, the line on the form to which error and other messages are sent. user-defined function key labels also reside in a buffer area while they are active. There is a buffer area for global labels and another for form-specific labels. These buffer areas are managed by VPLUS/V and are in addition to any buffers the application may define.

Form Definition Area

A form displayed on the terminal screen consists of protected areas (headings, labels, titles, display-only fields) and the unprotected areas (fields) into which data can be entered by a user.

Any form written to the terminal screen by VSHOWFORM already resides as a form image in memory. It may also be stored in terminal local form storage. The form image contains the description of all the protected areas on the form, except for "display only" fields. It also contains the visual enhancements for all fields, protected and unprotected. The form image can be loaded from the forms file into local form storage by VLOADFORMS. Or the form image is read from the forms file into memory by a call to VGETNEXTFORM. Then it is written to the terminal screen or loaded into local form storage depending on options that can be set in VREADFIELDS or VSHOWFORM.

Associated with each form is a set of data specifications -- field attributes and processing specifications defined in FORMSPEC. These specifications are read from the forms file by VGETNEXTFORM and reside in memory during execution of the form. The field attributes can be dynamically altered using VCHANGEFIELD.

Data Buffer Area

Besides the form definition, there is a data buffer area in memory that contains data for all the fields (unprotected and display only) defined for the form. These fields reside in the buffer in the order they are defined on the screen, from left to right, top to bottom.

When **ENTER** is pressed at the terminal, VREADFIELDS transfers the user-entered data from the screen to the data buffer. Before data is entered, if there are any initializations, VINITFORM sets the appropriate fields in the buffer to initial values. In the field phase, VFIELDEDITS verifies and possibly modifies the user-entered data in the data buffer according to any edit specifications defined for the fields. VFINISHFORM performs any final form modifications specified in the "finish" phase. Data for display-only fields may be written to the buffer by VINITFORM, VFIELDEDITS, or VFINISHFORM. Any changes to the data buffer are displayed at the terminal by the next VSHOWFORM.

An application can send data to the buffer with VPUTBUFFER or to a single field in the buffer with VPUTFIELD or VPUTtype. Conversely, the data buffer can be copied to an application with VGETBUFFER, or a single field can be copied with VGETFIELD or VGETtype.

COMMUNICATION AREA

Every application that calls VPLUS/V intrinsics must allocate a data area in the application for communication with the intrinsics. This area (called the *comarea*) is the first, and often the only, parameter in every call to a VPLUS/V intrinsic. Table 6-4 briefly outlines the contents of this communication area; Table 6-5 outlines the items used for data capture devices. It is essential to successful operation of your application that the *comarea* be defined exactly as shown in these tables; items that are "Reserved for system use" should not be set or used after they are initialized. The *comarea* items listed in table 6-4 and 6-5 are defined below.

Table 6-4. Outline of Communication Area Contents

Data Type	Position	Offset	Name	Function
Integer (Two-byte)	1	0	<i>cstatus</i>	status, error returns
	2	1	<i>language</i>	programming language of calling program
	3	2	<i>comarealen</i>	length of <i>comarea</i> in two-byte words
	4	3	<i>usrbuflen</i>	<i>comarea</i> extension length (BASIC)
	5	4	<i>cmode</i>	current mode (collect or browse)
	6	5	<i>lastkey</i>	code of last key pressed
	7	6	<i>numerrs</i>	number of errors in current form
	8	7	<i>windowenh</i>	code for window enhancement
	9	8	<i>multiusage</i>	Child of previous form
	10	9	<i>labeloption</i>	function key label indicator
Character Array (16-byte)	11-18	10-17	<i>cfname</i>	current form name (15 characters)
	19-26	18-25	<i>nfname</i>	next form name (15 characters)
Integer (Two-byte)	27	26	<i>repeatapp</i>	current form flag (repeat/append)
	28	27	<i>freezapp</i>	next form flag (freeze/append)
	29	28	<i>cfnumlines</i>	number of lines in current form
	30	29	<i>dbuflen</i>	data buffer length (in bytes)
Integer (Two-byte)	31	30	Reserved for system use.	--

Table 6-4. Outline of Communication Area Contents, continued

Data Type	Position	Offset	Name	Function
Logical (Two-byte)	32	31	<i>lookahead</i>	flag for background form load in VREADFIELDS
	33	32	<i>deleteflag</i>	delete current batch record
	34	33	<i>showcontrol</i>	control flags for VSHOWFORM
Integer (Two-byte)	35	34	Reserved for system use.	--
Integer (Two-byte)	36	35	<i>printfilnum</i>	MPE file number of forms file print file
	37	36	<i>filerrnum</i>	MPE file error number from FCHECK
	38	37	<i>errfilenum</i>	MPE file number, error message file
	39	38	<i>formstoresize</i>	number of forms in buffer
Integer (Two-byte)	40-42	39-41	Reserved for system use.	--
Integer (4-byte)	43-44	42-43	<i>numrecs</i>	number of records in batch file
	45-46	44-45	<i>recnum</i>	record # of current batch record
Integer (Two-byte)	47-48	46-47	Reserved for system use	--
Logical (Two-byte)	49	48	<i>filen</i>	MPE file number of terminal
Integer (Two-byte)	50-54	49-53	Reserved for system use	--
Logical (Two-byte)	55	54	<i>retries*</i>	max number of retries
	56	55	<i>termoptions*</i>	suppress msgs. and autoread
	57	56	<i>environ</i>	term environment: Sys LDEV
	58	57	<i>usertime*</i>	user defined time out period in seconds.

* Not supported on data capture devices.

Table 6-4. Outline of Communication Area Contents, continued

Data Type	Position	Offset	Name	Function
Logical (Two-byte)	59	58	<i>identifier</i>	type of terminal
	60	59	<i>labinfo</i>	number of function keys; length
Integer (Two-byte)	61-64	60-63	Reserved for system use	--
	65	64	<i>buffercontrol</i>	control ARB processing
	66	65	<i>bufferstatus</i>	indicate successful data conversion
	68-70	67-69	Reserved for system use	--
Refer to Table 6-5.	71-85	70-84	--	These <i>comarea</i> items are only referenced when using data capture devices.

Table 6-5. Communication Area Contents for Data Capture Devices

Data Type	Position	Offset	Name	Function
Integer (Two-byte)	71	70	<i>numflds</i>	number of fields on current form
	72	71	<i>splitpause</i>	length of pause (in seconds)
	73	72	<i>leftmodule</i>	type of terminal options
	74	73	<i>rightmodule</i>	type of terminal options
	75	74	<i>keyboard</i>	type of keyboard
	76	75	<i>display</i>	type of display
	77	76	<i>keyboardover</i>	whether to override or not
Character (Two-byte)	78	77	<i>errorlight</i>	error detection light
Logical (Two-byte)	79-80	78-79	<i>userlightson</i>	Lights on/off indicator
Integer (Two-byte)	81-85	80-84	Reserved for system use	--

NOTE

The *comarea* size required for each feature can be summarized as follows:

FEATURE	COMAREALEN
Data Capture Terminal	85
ARB Feature	70
All Others	60

The position of each item in *comarea* is given as a two-byte word offset in the table; for SPL programs the word offset starts with zero, for all other programs it starts with one.

The *comarea* must be at least 60 two-byte words long (120 bytes) unless you are working with an ARB. To take advantage of the ARB feature, the *comarea* must be at least 70 words long. For BASIC programs, the area must be extended to include space for the form and data buffers and for internal tables. For non-BASIC programs, the DL area is used for these buffers and internal tables, and the communication area need not be extended. However, non-BASIC programs must be careful not to use the DL area for other purposes when using VPLUS/V. Refer to Appendix E for more information on the DL area.

Before calling the first VPLUS/V intrinsic, you should initialize the entire *comarea* to binary zeroes. Do not change *comarea* values between calls except under documented conditions. Then, if you are coding in a programming language other than COBOL, you must set *language* to the code for the programming language you are using. If you are coding in BASIC, you must set *usrbuflen* to the number of two-byte words needed for the *comarea* extension. This value is displayed on the forms file listing generated by FORMSPEC. Finally, in order to provide for possible future extensions to *comarea*, set *comarealen* to its current total length.

Parameters

cstatus Two-byte integer to which the intrinsic status is returned. Set to zero if the call is successful; to a nonzero value if an error occurs. If the error is an MPE file error, a file error number is also returned to *filerrnum*. (Refer to Appendix B for a list of the error codes that may be returned to *cstatus* with their meaning.) It is up to the user to provide error-handling routines and to reset *cstatus*.

language Two-byte integer that indicates the programming language of the calling program:

- 0 = COBOL II/III
- 1 = BASIC
- 2 = FORTRAN 66
- 3 = SPL
- 5 = PASCAL/HP FORTRAN 77/HP BUSINESS BASIC

The *language* must be set by the calling program before any intrinsic is called.

comarealen

Two-byte integer that indicates the total length of *comarea* (use 60 for terminals; 85 for data capture devices). The *comarealen* should be specified for each *comarea* to simplify future changes to the length of *comarea*.

usrbuflen

For BASIC programs only, *usrbuflen* must be set to the number of two-byte words to be appended to *comarea* for the form and data buffers and internal tables. (Note that the number of bytes required for this extension is printed when a form is listed through FORMSPEC.) The *usrbuflen* need not be set by other languages since VPLUS/V automatically uses the DL area for buffers and internal tables. The *usrbuflen* does not include the *comarea* length specified in *comarealen*, but it must be contiguous to and immediately follow *comarea*.

cmode

Indicates whether current mode of data entry is collect.

0 = Collect Mode

1 = Browse Mode

If you want to save the form sequence for \$RETURN, you must set *cmode* to zero before calling VGETNEXTFORM.

- 0 = 0 Do not enable the touch feature (default).
- 1 Enable the touch feature.

Bits 15, 14, and 13 are used to override VSHOWFORM optimization (where only information that has changed since the last call to VSHOWFORM is written to the terminal screen). Bits 12 and 11 are used with VREADFIELDS on data capture devices. Bit 10 is used when consecutive calls are made to VSHOWFORM (no call to VREADFIELDS in between). Bit 9 is used with terminals having local form storage capability.

Bit 8 is used to allow access to the function control keys on the HP 239X and HP 262X terminals. For example, device control and device mode keys can be used for local printing to an integral printer. Enabling the function control keys using bit 8 is recommended for experienced users only. Misuse or accidental use of certain function control keys could yield undesirable results.

Bit 0 is used to activate touch available with terminals with the touch feature. VREADFIELDS returns *lastkey* with the field number of the touched field (as a negative number).

printfilnum MPE file number to which a form is printed by VPRINTFORM. The formal designator is FORMLIST.

filerrnum MPE file error number (FCHECK number) returned by VPLUS/V intrinsics when an MPE file error occurs. (Refer to Appendix B for a list of the errors that return a number to *filerrnum*.)

errfilenum MPE file number of the error message file used by VERRMSG.

formstoresize Two-byte integer that indicates the number of entries allowed in the form storage directory.

- 1 No local form storage is to be performed. For the HP 2626A terminal the workspace configuration is under user control. VPLUS/V does not modify terminal workspace, window, or datacomm configuration, and either terminal port can be used.
- 0 No local form storage is to be performed. For the HP 2626A terminal VPLUS/V reconfigures the workspaces and windows to the terminal's default values, but only the first terminal port can be used.
- 1..255 One through 255 forms can be stored locally. For the HP 2626A terminal any value greater than four defaults to four (allowing an application to run on either the HP 2626A, HP 2394A, or HP 2624B terminals), where four indicates that one form is the displayed form and three forms are stored in the HP 2626A local forms storage. VPLUS/V reconfigures the HP 2626A terminal workspaces and windows to support local form storage, but only the first terminal port can be used.

The *formstoresize* allows the user to control the number of entries allowed in the form storage directory on the user stack and must be set prior to opening the terminal and the forms file. The space required is listed on the forms file listing generated by FORMSPEC.

When designing forms for local form storage terminals, keep in mind that large forms need a substantial amount of terminal display memory. For forms that are greater than 24 lines, so much terminal display memory may be used that insufficient terminal memory is available for form storage. If so, VPLUS/V resets *formstoresize* to zero and no local form storage is performed.

numrecs

Four-byte integer that contains the number of nondeleted records in the current batch file. It is used by the V..BATCH intrinsic.

recnum

Four-byte integer set to the current record number in the batch file. (Note that record numbers start with zero.) The *recnum* must be set by the program before writing to or reading from the batch file. It is used by the VREADBATCH and VWRITEBATCH intrinsics.

filen

MPE file number used to identify the terminal.

retries

Maximum number of retries.

value = 0 Use default value (four retries)
value > 0 Use this value as maximum
value < 0 do not perform any retries

termoptions

Terminal control options:

bits: 0-8	reserved for system use
9-10	01 = enables ENTER /function key timeout in VREADFIELDS. 11 or 00 = disables ENTER /function key timeout in VREADFIELDS. (default)
11-12	10 = do not clear the screen at terminal open or close.
13-14	01 = enables AUTOREAD in VREADFIELDS. 11 or 00 = disables AUTOREAD. (default)
	(AUTOREAD causes the terminal to do a programmatic enter instead of waiting for the user to press ENTER .)
15	0 = Sound bell as usual (default). 1 = Suppress bell.

NOTE

In programming languages which do not have the capability of setting single bits, programmers should add or subtract appropriate amounts to change single bits. For example, if *termoptions* was initialized entirely to zeros and the programmer wishes to enable the AUTOREAD, the following statement should be issued in COBOL:

ADD 2 TO TERM-OPTIONS.

Conversely, if disabling the AUTOREAD feature, the programmer should subtract 2 from *termoptions*. For more information on the AUTOREAD feature, please consult Appendix G.

<i>environ</i>	First byte is the logical device number of the terminal. The remaining byte is reserved for system use.
<i>usertime</i>	If enabled, the value in this position is used as the number of seconds to wait for either ENTER or a function key to be pressed. Consult Appendix G for more information regarding enabling user timeouts.
<i>identifier</i>	VPLUS/V identifier for the terminal model being used. See Appendix G for the interpretation of the contents of this word.
<i>labinfo</i>	First byte is the length of key labels (in bytes); Second byte is the number of key labels that VPLUS/V uses on the terminal.
<i>buffercontrol</i>	Controls the ARB options. The bit settings are as follows: Bit 15 (<i>ARB option</i>) 0 = return data "raw": no reordering or conversion. 1 = reorder and convert data according to ARB specifications. The remaining bits must be set to zero. Note that VGETBUFFER will not change these settings (see Section 6, VGETBUFFER, for special considerations).
<i>bufferstatus</i>	Return parameter indicating that VGET/PUTBUFFER performed the data conversion successfully (bit 15 in offset 65 will be set to 1).

NOTE

STATUS in the *COMAREA* will be non-zero if the conversion is not successful.

In addition to the above *comarea* descriptions, the data capture devices use the following items:

Parameters

splitpause

Length of time in seconds to pause between the presentation of lines of text on the single line alpha display. Default is 3 seconds.

- 1 - wait for user to press key
- 0 - do not pause
- >0 - pause specified number of seconds

leftmodule

MPE determines which, if any, module is present.

- 0 - no module
- 1 - printer
- 2 - multifunction reader
- 3 - RS232 interface
- 4 - typeV badge reader
- 5 - magstripe reader
- 6 - bar code reader
- 7 - HP-IB interface

rightmodule

MPE determines which, if any, module is present.

- 0 - no module
- 1 - printer
- 2 - multifunction reader
- 3 - RS232 interface
- 4 - typeV badge reader
- 5 - magstripe reader
- 6 - bar code reader
- 7 - HP-IB interface

NOTE

VPLUS/V does not communicate with the RS-232 interface (3) or the HP-IB interface (7).

keyboard

Type of keyboard used with terminal.

- 0 - HP 3077-no keyboard.
- 1 - standard keyboard (12 function keys with values of -1,0,17...26).
- 2 - alphanumeric keyboard (28 function keys with values of -1,0...26).

display

Indicates the terminal type of display.

- 0 - numeric display
- 1 - alphanumeric display
- 2 - mini-CRT display

VPLUS/V does not support the terminal that has numeric display.

keyboardover

Two-byte integer determining whether to override input on the keyboard or not. Default is 0.

- 1 - Override and enable the keyboard without regard to forms design.
- 0 - Do not override. Allow input from the devices specified during the form design (in the Processing Specification area).

VGETBUFFER

Copies entire contents of data buffer from memory to an application.

Syntax

VGETBUFFER {*comarea*,*buffer*,*buflen*}

Parameters

comarea Must be *comarea* name specified when the forms file was opened with VOPENFORMF. If not already set, the following *comarea* items must be set before calling VGETBUFFER:

cstatus Set to zero.

language Set to the code identifying the programming language of the calling program.

comarealen Set to total number of two-byte words in *comarea*. Must be at least 70 words in length if the ARB feature is used.

buffercontrol Set bit 15 to 1 to indicate that data is to be transformed according to the ARB specifications.

VGETBUFFER may set the following *comarea* items:

cstatus Set to nonzero value if call unsuccessful.

bufferstatus Bit 15 set to 1 if data conversion successful.

buffer Character string in an application to which the data in the data buffer is copied. Could also be a record describing the ARB, i.e., non-CHAR data.

buflen Two-byte integer variable that specifies the number of bytes to be transferred to the user buffer.

Discussion

This intrinsic transfers data from the data buffer in memory to the area in an application specified by the *buffer* parameter. The data includes everything in the unprotected and display-only fields on a form. Previously, data in the data buffer was stored in the order of the fields on the form, and specific fields could be moved from the buffer with VGETFIELD or VGETtype. Now, the Application-ready Buffer (ARB) allows you to specify, using the FORMSPEC ARB feature, the order in which the application should receive the fields in the buffer, and the data type conversion to be performed on each (see the discussion on creating an ARB in Section 3.) You need no longer use VGETtype to convert individual fields; a call to VGETBUFFER accomplishes the task.

VGETBUFFER

The number of bytes moved from the data buffer is based on the number of bytes specified in the *buflen* parameter, or by the *dbuflen* in the *comarea* (refer to Table 6-5), whichever is less. The *dbuflen* item is set by VGETNEXTFORM when the current form is read into memory. For example, if there are 20 bytes in the data buffer (*dbuflen* is 20), and the user requests 50 bytes in the *buflen* parameter, only 20 bytes are transferred. Conversely, if the user requests 10 bytes through the *buflen* parameter, and *dbuflen* is 20, only 10 bytes are transferred.

Special Considerations

Designers using the ARB feature in VGETBUFFER should be aware that damaging run-time errors could occur if the application is inadvertently run on a system that has a VPLUS/V version earlier than B.05.00.

To prevent this, the designer should do three things:

1. Document the product with a clear warning that VPLUS/V version B.05.00 or later **MUST** be used.
2. Use the VPLUS/V intrinsic HP32209 in the code. This intrinsic checks to make sure that you are using the proper VPLUS/V version. If not, the application should terminate with an appropriate message.
3. Check offset 65 (*bufferstatus*) in the *comarea* on return from VGETBUFFER. Bit 15 will be set to 1 if VGETBUFFER performed the conversion successfully. In other words, the application must check both *status* and *bufferstatus* to be sure that the data was correctly converted.

Example

COBOL

01 ORDER-ENTRY.

03 PART-NO PIC X(7).
03 DESCR PIC X(12).
03 QTY PIC S9(4).
03 UNIT-PR PIC S9(4)V9(2).
03 TOTL-PR PIC S9(6)V9(2).

CALL "VGETBUFFER" USING COMAREA, ORDER-ENTRY, DBUFLEN.

BASIC

340 L1=D1 ← D1 is the *dbuflen* word in C
350 CALL VGETBUFFER(C(*),D\$,L1)

FORTRAN

CALL VGETBUFFER(COMAREA,D1,DBFLEN)

SPL

BYTE ARRAY D1(0:36);

VGETBUFFER(COMAREA,D1,DBUFLEN);

These examples transfer the contents of the data buffer in memory to an application. The current value of *dbuflen* is specified as the user buffer length.

VGETFIELDINFO

Discussion

This intrinsic accesses an internal table and places information about one or more fields into *infobuf*. You tell VGETFIELDINFO how many fields, how much information about each field, and the name of the form containing the fields.

Table 6-10. Field Information Buffer

Data Type	Position	Contents	Comments
Integer (Two-byte)	1	<i>numofentries</i>	Required
	2	<i>entrylength</i>	Required
Character Array (16-byte)	3-10	<i>formname</i>	Required; second byte of position 10 is unused
	11-18	Field name	Permissible key; last byte of position 18 is unused
Integer (Two-byte)	19	Field number according to screen order.	Permissible key
	20	Field number	Permissible key; in order of creation
	21	Field length	In bytes
	22	Position of field in data buffer	In bytes, offset from zero
Character Array (4-byte)	23-24	Field enhancement	Combination of I, H, U, B, 1-8, or NONE.
	25-26	Data type of field	May be CHAR, DIG, IMP <i>n</i> , date (MDY, DMY, YMD), or NUM[<i>n</i>].
Character (Two-byte)	27	Field type	First byte of position 27; may be O, R, P or D.

Passing INFOBUF Without Entering Keys

Keys are optional. If you do not supply a key to indicate the first field you want information about, VGETFIELDINFO starts with the first field not already reported on by the current call.

As an example, suppose you want information about 10 fields, and you enter screen order numbers 3, 5, and 6 as keys. VGETFIELDINFO returns information about fields 3, 5, 6, and 7 through 13. If you enter no keys, VGETFIELDINFO returns information about fields 1 through 10.

VGETFIELDINFO

Table Wraparound

VGETFIELDINFO starts over with the first field in the form if you request information about more fields than are in the form or, if when you pass a key, your request goes beyond the end of the form.

Suppose a form has 12 fields. If you request information about 10 of the fields and enter screen order number 8 as a key, VGETFIELDINFO returns information about fields 8 through 12, and then goes to the beginning of the form and returns information about fields 1 through 5.

Example

```
COBOL
DATA DIVISION.
:
WORKING-STORAGE SECTION.
01  INFOBUF.
    05  NUMBER-OF-ENTRIES                PIC S9(4) COMP.
    05  ENTRY-LENGTH                    PIC S9(4) COMP.
    05  FORM-NAME                        PIC X(15).
    05  FILLER                           PIC X.
    05  ENTRY-TABLE OCCURS 10 TIMES.
        10  FIELD-NAME                  PIC X(15).
        10  FILLER                      PIC X.
        10  SCREEN-ORD-NUM              PIC S9(4) COMP.
        10  FIELD-NUM                  PIC S9(4) COMP.
        10  FIELD-LENGTH                PIC S9(4) COMP.
        10  FIELD-POSITION              PIC S9(4) COMP.
        10  FIELD-ENHANCE               PIC X(4).
01  INFOBUFLEN                          PIC S9(4) COMP VALUE 80.
:
PROCEDURE DIVISION.
:
    MOVE SPACES TO INFOBUF.
:
    MOVE 5 TO NUMBER-OF-ENTRIES.
    MOVE 14 TO ENTRY-LENGTH.
    MOVE "FORM1" TO FORM-NAME.
    MOVE 2 TO SCREEN-ORD-NUM(1).
    CALL "VGETFIELDINFO" USING COMAREA, INFOBUF, INFOBUFLEN.
```

The example shown above illustrates the data declaration of *infobuf* and *infobuflen* and the passing of parameters to VGETFIELDINFO. Note that before the intrinsic is called, *infobuf* is initialized to spaces (*not* zeros). The intrinsic copies 14 two-byte words of information about fields 2 through 6 of form FORM1 into *infobuf*.

VGETNEXTFORM

multiusage

Set to 1 if current form is child or sibling of previous form, zero otherwise.

In addition, if a new form has been retrieved (*repeatapp* is zero), VGETNEXTFORM sets the following items:

<i>cfnmlines</i>	Set to number of lines in form just read (now the current form).
<i>nfname</i>	Set to name of next form to be read from file. See Discussion below for more information.
<i>repeatapp</i>	Set to value read from forms file for this (current) form.
<i>freezapp</i>	Set to value read from forms file for this (current) form.
<i>dbuflen</i>	Set to length (in bytes) of the current form just read from the forms file.

Discussion

VGETNEXTFORM checks the value of *repeatapp* passed in *comarea*. If this value indicates the current form is to be repeated, or repeated and appended to itself, it does not read the next form, nor does it update the values of *cfnmlines*, *nfname*, *repeatapp*, *freezapp*, or *dbuflen*. Note that a repeating form is repeated until *repeatapp* is set to zero, either by an application or, for ENTRY, when the user presses **NEXT FORM** to request the next form, or by the FORMSPEC processing language. If the current form is not to be repeated, VGETNEXTFORM checks *nfname* to determine which form to read from the forms file.

Example

```
COBOL
CALL "VGETNEXTFORM" USING COMAREA.
```

```
BASIC
11 CALL VGETNEXTFORM(C(*))
```

```
FORTRAN
CALL VGETNEXTFORM(COMAREA)
```

```
SPL/PASCAL
VGETNEXTFORM(COMAREA);
```

The examples above call VGETNEXTFORM to retrieve the next form from the forms file and reset the *comarea* according to the values in the next form.

VGETNEXTFORM

Copies character coded data contents from data buffer into an application, converting numeric value to specified type.

Syntax

`VGETtype {comarea,fieldnum,variable} [{,numdigits,decplaces}] *`

Parameters

type

The *type* in VGETtype indicates that this intrinsic may be specified as:

VGETINT	converts value to two-byte integer
VGETDINT	converts value to four-byte integer
VGETREAL	converts value to four-byte real value
VGETLONG	converts value to eight-byte long value
*VGETPACKED	converts value to packed decimal format (COBOL <i>usage is comp-3</i> format); this intrinsic has two extra parameters, <i>numdigits</i> and <i>decplaces</i> . Both are two-byte integer variables that contain the number of digits and number of decimal places, respectively, specified by the COBOL <i>usage is comp-3</i> data declaration.
*VGETZONED	converts value to zoned decimal format (COBOL default format); has the parameters <i>numdigits</i> and <i>decplaces</i> , which are two-byte integer variables that contain the number of digits and number of decimal places, respectively, specified by the COBOL data declaration.
VGETYYMMDD	converts date to a six-byte character value

For example: CALL "VGETINT" USING COMAREA, FIELDNUM, VARIABLE

CALL "VGETPACKED" USING
COMAREA, FIELDNUM, VARIABLE, NUMDIGITS, DECPLACES

comarea

Must be *comarea* name specified when the forms file was opened with VOPENFORMF.

If not already set, the following *comarea* items must be set before calling VGETtype:

cstatus Set to zero.

comarealen Set to total number of two-byte words in *comarea*.

VGETtype may set the following *comarea* items:

cstatus Set to nonzero value if call unsuccessful or if requested field has an error.

numerrs May be decremented if new value replaces the value of a field that was in error.

fieldnum Two-byte integer variable containing the number assigned to the field by FORMSPEC.

variable Variable within application, of type specified in VGETtype, into which converted value is placed.



Discussion

If there is no ARB, all data is read from the unprotected fields on the screen as character strings and concatenated to create a data buffer. The VGETBUFFER intrinsic copies the entire buffer to an application program or VGETFIELD can be used to obtain the contents of an individual field.

NOTE

If you are using VGETBUFFER in conjunction with an ARB, you do not need to use VGETtype. VGETBUFFER performs all the required conversions on the screen data in the buffer.

You can use both VGETBUFFER with an ARB and VGETtype calls in the same program: the *buffercontrol* setting in the *comarea* that controls ARB processing can be switched on or off for each form.

The VGETtype intrinsics, VGETINT, VGETDINT, VGETREAL, VGETLONG, VGETYYMMDD, VGETPACKED, and VGETZONED, have been provided to perform conversion from character coded data to the seven indicated data types. This intrinsic copies the contents of the field identified by its field number from the data buffer. (Note that this field number is a unique number assigned to each field by FORMSPEC and is totally independent of the field position in the data buffer.) The field's value must be numeric, but its data type need not be. That is, numbers in a character type field can be converted.

The numeric value, stored in the buffer in character coded form, is converted to the specified type and then copied to the variable in the application. (Refer to table 6-13 for the format of each type.) If errors occur during conversion, *cstatus* is set to an error code. If the requested field has an error, its value is moved to the variable, but *cstatus* is set to an error code.

VGETtype

Table 6-13. Numeric Type Conversion

Type	Format
INT	Two-byte word fixed-point; twos complement representation of positive and negative values; range from -32768 through +32767.
DINT	Four-byte word fixed-point format; twos complement representation of positive and negative range between approximately -2 billion and +2 billion. (Not used by BASIC programs.)
REAL	Four-byte word floating-point format with sign bit in bit 0; an exponent (biased by +256) in bits 1 through 9, and a positive fraction in the remaining 22 bits in HP 3000 format. (Not used by COBOL programs.)
LONG	Floating-point format using eight-bytes; sign bit in bit 0; an exponent (biased by +256) in bits 1 through 9, and a positive fraction in the remaining 54 bits in HP 3000 format. (Not used by COBOL programs.)
PACKED	COBOL only; <i>comp-3</i> . See COBOL II/3000 Reference Manual.
ZONED	COBOL only. See COBOL II/3000 Reference Manual.
YYMMDD	Six-byte ASCII, in format YYMMDD; for example, 870623.

VGETtype

The following chart will help you correlate VPLUS/V data types with VGETtype intrinsics and programming language data types.

Table 6-14. Correlation of VGETtype with the Data Types for Each Language

Language	Intrinsic	Data Type
FORTRAN	VGETINT VGETDINT VGETREAL VGETLONG	Integer*2 Integer*4 Real Double Precision
BASIC	VGETINT VGETREAL VGETLONG	Integer Real Long
SPL	VGETINT VGETDINT VGETREAL VGETLONG	Integer Double Real Long
COBOL	VGETINT VGETDINT	S9-S9(4)COMP S9(5) - S9(9)COMP
PASCAL	VGETINT VGETDINT VGETREAL VGETLONG	Subrange -32768..32767 Integer Real Longreal

1. If errors occur during conversion, the two-byte status word in the communications area is set to an error value.
2. If the requested field has been flagged as having an error (perhaps by VFIELDEDITS or VSETERROR), the conversion is performed, but the two-byte status word is also set to an error value.
3. An attempt to convert a number larger than 32767 using VGETINT returns an error value (504) in the two-byte status word and leaves the receiving value unchanged.
4. All numeric separators are stripped before conversion is performed.
5. Fields of type CHAR may be converted as long as they contain numeric characters (including signs and separators). Otherwise, an error value is returned in the two-byte status word.
6. VGETINT and VGETDINT only convert the integer portion of a given field. The fractional portion is truncated before conversion. Remember that in a field of type IMPn, the rightmost "n" characters are treated as a fraction.
7. Note that if VGETFIELD is used to pass a field containing a decimal point to a COBOL program, the decimal point is also passed and no arithmetic may be performed on the field.

VPRINTFORM

Discussion

VPRINTFORM prints the current form to a list file. It is analogous to VSHOWFORM, in that it prints the form and the current data buffer values, except that VPRINTFORM prints the form on a hardcopy device rather than on the terminal. Enhancements obviously cannot be shown directly; the window line and key labels are not printed. The form must have been read into the form definition area of memory by a prior call to VGETNEXTFORM. The information printed depends on what is in the data buffer, which may not necessarily be the same as what is displayed.

The carriage control character specified in the *pagecntl* parameter is effective after the form is printed.

If the calling program opens the list file, it must supply the file number of this file in *printfilnum*. If *printfilnum* is zero, VPRINTFORM opens a list file and sets *printfilnum* to the file number of the file. VPRINTFORM opens the list file, with the formal and actual file designator FORMLIST, assigns it to the device class LP, and specifies its length as 80 bytes. This is equivalent to using the file equation:

```
:FILE FORMLIST;DEV=LP;REC=-80
```

A user may change any of these characteristics with an MPE :FILE command.

Example

```
COBOL  
CALL "VPRINTFORM" USING COMAREA, UNDERLINE, PAGE.
```

```
BASIC  
135 CALL VPRINTFORM(C(*),U,P)
```

```
FORTRAN  
CALL VPRINTFORM(COMAREA,UNDRLN,PAGE)
```

```
SPL/PASCAL  
VPRINTFORM(COMAREA,UNDERLINE,PAGE);
```

Each of the calls shown above prints the current form on a list device; if not already open, it opens the device file.

VPRINTSCREEN

Prints the entire contents of a terminal screen to an offline list device or laser printer during VPLUS execution.

Syntax

VPRINTSCREEN {*comarea*, *readsize*}

Parameters

comarea If not already set, the following *comarea* items must be set before calling VPRINTSCREEN:

<i>cstatus</i>	Set to zero.
<i>comarealen</i>	Set to the total number of two-byte words in <i>comarea</i> .
<i>printfilenum</i>	Set to the file number of the list file to which the form is printed. If set to 0, VPRINTSCREEN opens the device "LP" as the list file and sets <i>printfilenum</i> to the file number of the opened list file.

VPRINTSCREEN may set the following *comarea* items:

<i>printfilenum</i>	If VPRINTSCREEN opened the list file, set to the file number of the opened file.
<i>cstatus</i>	Set to a nonzero value if call is unsuccessful.
<i>fileerrnum</i>	Set to the MPE error code if an MPE file error is encountered.

readsize Two-byte integer, reserved for system use. Must be set to 0.

Discussion

This intrinsic provides the capability for printing the current screen display with function keys, line drawing characters and appended forms, as well as the data on the screen. It differs from VPRINTFORM, which is limited to printing only the data in the form data buffer. However, VPRINTFORM offers the option of underlining fields, which VPRINTSCREEN in LP mode does not.

The programmer can produce copies of VPLUS screens in either of two ways: incorporating VPRINTSCREEN into an application so that screen images can be captured at run-time with their data, or developing a simple utility that allows data to be entered into the screens before calling VPRINTSCREEN to reproduce them. The utility has the advantage of removing the overhead caused by

VPRINTSCREEN

VPRINTSCREEN from the application, while still providing a way to reproduce screens and data for product literature.

VPRINTSCREEN always uses the Pascal HEAP procedures to perform stack allocation. This introduces the risk of conflict for programs written in COBOL, FORTRAN/66 or SPL, because the existing intrinsics would use DLSIZE for stack allocation in these cases. Applications written in these languages must, therefore, follow two rules when calling VPRINTSCREEN. They are:

1. The language id in the *comarea* must be set to 5.
2. The INTRINSIC calling mechanism must be used when calling the VPLUS intrinsics from the main and all interacting parts of the application.
For example, with COBOL the application must use:

```
:CALL INTRINSIC <intrinsic name>
```

Refer to Appendix E of this manual, and to the COBOL and Pascal reference manuals, for more information on calling mechanisms.

For applications that use a language id of 5 in the VPLUS *comarea*, including Pascal, FORTRAN/77 and HPBUSINESS BASIC, VPLUS uses the HEAP procedures for stack allocation, so VPRINTSCREEN can be called in the standard format.

The programmer may implement this feature by defining a function key that allows the user to print the screen contents at any time. This would be useful for providing immediate output during production.

Modes of Operation

VPRINTSCREEN operates in two modes, normal and documentation mode. A VPLUS-supported terminal is required for execution of this intrinsic.

Normal Mode

This is the default calling mode for VPRINTSCREEN. When called, the value in the *printnumfile* word of the *comarea* is used to determine the list device. If the calling program opens the list file, it must supply the file number of this file in *printfilenum*. VPRINTSCREEN opens the list file with the formal and actual file designator FORMLIST, assigns it to the device class LP, and specifies its length as 80 characters. This is equivalent to using the file equation:

```
:FILE FORMLIST;DEV=LP;REC=-80
```

The user may change any of these characteristics with a :FILE command.

NOTE

It is recommended that VPRINTSCREEN and VPRINTFORM not be used in the same program. Since the same list file is used for both listings, output from the two calls will be intermixed.

Each time VPRINTSCREEN is called, a PAGE EJECT is performed at the end of the print operation.

VPRINTSCREEN

Documentation Mode

You require TDP in order to use VPRINTSCREEN in this mode. VPRINTSCREEN, in conjunction with TDP, provides the capability to print screen contents on a laser printer (HP2680A or HP2688A). In this mode, field highlighting other than color, borders, alternate character sets and active function keys are captured and converted to the requisite font for printing on the laser printer.

Documentation mode is enabled by setting a JCW before running the program. It is:

```
:SETJCW VPRINTJCW=1
```

When VPRINTJCW is set to 1, the list file FORMLIST is NOT opened. Instead, a temporary file called EPOCLIST is created (or appended to, if it already exists). EPOCLIST can be saved and renamed on completion of the screen capture, then input to TDP and 'finald'. The user can add text to the file or include it in a separate TDP file. Refer to the *TDP Reference Manual* (Part No. 36578-90004) for more information on use and include files.

In order to print the forms on a laser printer, an environment file must be created (if it does not already exist). Refer to the *IFS/3000 Reference Guide* (Part No. 36580-90001) for more information. The environment files for the HP2680 and HP2688 laser printers are VENV80 and VENV88 respectively. All environment files must include the font ids listed under "Limitations" below.

EPOCLIST uses VSETUP, a TDP include file, as the default file to reference the environment files, which must be accessible to TDP in order for EPOCLIST to be printed. If an environment file other than VENV80 or VENV88 is used, EPOCLIST must be modified to reference this file, and the font definitions from VENV80 and VENV88 must be included in it (see "Limitations" below).

Printing Screens from TDP. To print the contents of EPOCLIST to a laser printer, follow these steps:

1. :PURGE (or :RENAME) EPOCLIST
2. :SETJCW VPRINTJCW=1
3. :RUN <your application program>
4. Use the print function wherever it is available to save the screen contents, including data, in EPOCLIST.
5. Exit the application.
6. :SAVE EPOCLIST
7. :RENAME EPOCLIST, <new name> If you do not rename EPOCLIST, the output generated the next time the application is run will be appended to the existing file.
8. 6fd:RUN TDP.PUB.SYS
9. Text in the file and execute the command, "Final from <new name> to *HP2680" (or HP2688).

The screens will be printed out, one per page. Steps 8 and 9 may be specified in a job stream.

Merging Screens with a TDP File. You can include screens in an existing TDP document easily by using the file VEPOCUSE, a TDP use file that divides EPOCLIST into separate files containing one screen per file. The VSETUP file must be included as one of the first statements of the TDP document so that the correct environment file is referenced for printing the screens.

Follow these steps:

1. :RUN TDP Clear the workspace.
2. :USE VEPOCUSE Answer the prompts, and enter a PREFIX. VEPOCUSE separates the old file into new files, each containing one screen.
3. Text in the TDP document and include each screen file in the correct place.

It is recommended that screen files be included as separate, individual files in a document, rather than being incorporated directly into the text, because EPOCLIST has a record size of 168 bytes and most document files are set to 80 bytes.

Limitations

The following limitations pertain to the use of VPRINTSCREEN.

- Procedure calls must be modified in order to call the intrinsic from a language that does not use a VPLUS language id of 5 in the *comarea*; for example, COBOL, SPL and FORTRAN/66.
- It uses additional stack resources.
- Native Language support is NOT available for VPRINTSCREEN. To print a screen in another language, you require a LOCALIZED environment file which maps to the following fonts:

- c = full bright
 - g = half-bright inverse video
 - d = normal
 - l = line-draw
 - m = math

- The current environment files do not distinguish between full bright and half bright, but the code is set up to do so.
- Screen images cannot be scaled; fonts come in one size only.
- The maximum TDP record size is 168 characters. A single line in a screen may easily exceed this limit if it contains many escape sequences, in which case the line may be truncated.
- TDP macros 5-9 are used to minimize truncation, but some screens will reach the limit anyway. If you use your own macros 5-9, some inconsistencies may appear when text and screens are merged.
- TDP may indicate that errors have occurred when, in fact, there are none. The most common error messages are: "Unrecognizable command" and "n characters have been truncated".

VPRINTSCREEN

Example

This is a sample EPOCLIST file.

```
*>>DATE: FRI, MAY 3, 1987, 11:19AM
if main in hpvsetup.pub
image 28
need 28
>>>>>screen formatting commands<<<<<<
*>>END VPRINTSCREEN B.05.00
```

SAMPLE SCREEN v.FF

This is a sample form to demonstrate the VPRINTSCREEN intrinsic.

Name: []

Street Address: []

City: [] State: []

Phone: []

ADD

CHANGE

DELETE

REVIEW

HELP

PRINT

EXIT

Figure 6-4. Sample EPOCLIST Screen from TDP

VPUTBUFFER

Copies data from an application to the data buffer in memory.

Syntax

VPUTBUFFER {*comarea*,*buffer*,*buflen*}

Parameters

comarea

Must be *comarea* name specified when the forms file was opened with VOPENFORMF. If not already set, the following *comarea* items must be set before calling VPUTBUFFER:

cstatus Set to zero

language Set to the code identifying the programming language of the calling program.

comarealen Set to total number of two-byte words in *comarea*. Must be at least 70 words in length if the ARB feature is used.

buffercontrol Set bit 15 to 1 to indicate that data is to be transformed according to the ARB specifications.

VPUTBUFFER may set the following *comarea* items:

cstatus Set to nonzero value if call unsuccessful.

numerrs May be decremented as a result of new data replacing data in field with error.

bufferstatus Bit 15 set to 1 if data conversion is successful.

buffer

Character string in an application containing the data to be copied to the data buffer in memory.

buflen

Two-byte integer variable that specifies the number of bytes to be copied to the data buffer in memory. The *comarea* item *dbuflen*, which contains the size of the data buffer in memory, may be used as this parameter.

Discussion

This intrinsic transfers data from a buffer in an application to the data buffer in memory. The length of the data moved is based on the number of bytes specified in the *buflen* parameter and the number of bytes in the largest data buffer in the forms file, whichever is less. The length of the buffer assigned to

VPUTBUFFER

the current form is not considered since the user may intend the data for another form with a longer buffer length. For example, assume there are three forms in the forms file:

Form A *dbuflen* = 100 bytes
Form B *dbuflen* = 200 bytes
Form C *dbuflen* = 75 bytes

In this case, the maximum data buffer length is 200 bytes. If the current form is form A and the user calls VPUTBUFFER with a user buffer length (*buflen* parameter) of 200, he may intend to call VGETNEXTFORM to get form B and then VSHOWFORM to display form B with the 200 bytes of data moved to the data buffer with VPUTBUFFER.

Fewer bytes than the data buffer can hold may be transferred; the remaining space in the data buffer is not changed.

Previously, the data was moved to the data buffer exactly as it appeared in the application buffer, and you had to use VPUTtype to convert data in each field from the application data type to USASCII. Now, you can specify, using the FORMSPEC ARB feature, the mapping between the fields in the application buffer and those on the screen (see the discussion on creating an ARB in Section 3). Screen fields may be in a different order from those in the application buffer, and the data conversions will be performed when the data is written out to the screen. If any field being replaced by user data contained an error, VPUTBUFFER clears the error flag for the field and decrements *numerrs*.

Special Considerations

Designers using the ARB feature in VPUTBUFFER should be aware that damaging run-time errors could occur if the application is inadvertently run on a system that has a VPLUS/V version earlier than B.05.00.

To prevent this, the designer should do three things:

1. Document the product with a clear warning that VPLUS/V version B.05.00 or later **MUST** be used.
2. Use the VPLUS/V intrinsic HP32209 in the code. This intrinsic checks to make sure you are using the proper VPLUS/V version. If not, the application should terminate with an appropriate message.
3. Check offset 65 (*bufferstatus*) in the *comarea* on return from VPUTBUFFER. Bit 15 will be set to 1 if VPUTBUFFER performed the conversion successfully. In other words, the application must check both *status* and *bufferstatus* to be sure that the data was correctly converted.

Example

```
COBOL
01  DAT1.
    03  FIRSTNAME PIC X(6).
    03  LASTNAME  PIC X(18).
    .
ACCEPT DAT1.
CALL "VPUTBUFFER" USING COMAREA, DAT1, DBUFLEN.
```

VPUTBUFFER

BASIC

```
235 L1=24
```

```
240 CALL VPUTBUFFER(C(*),D1$,L1)
```

FORTRAN

```
CALL VPUTBUFFER(COMAREA,DAT1,DBLEN)
```

SPL/PASCAL

```
BYTE ARRAY DAT1(0:23);
```

```
⋮
```

```
VPUTBUFFER(COM1,DAT1,LEN);
```

The following calls transfer 24 bytes from an application area, DAT1 to the data buffer. In this example, the longest *dbuflen* is assumed to be 80 bytes.

VPUTBUFFER

VPUTFIELD

Copies data from an application into a specified field in the data buffer in memory.

Syntax

VPUTFIELD {*comarea*,*fieldnum*,*fieldbuf*,*buflen*,*actualen*,*nextfldnum*}

Parameters

<i>comarea</i>	Must be <i>comarea</i> name specified when the forms file was opened with VOPENFORMF. If not already set, the following <i>comarea</i> items must be set before calling VPUTFIELD:
<i>cstatus</i>	Set to zero.
<i>language</i>	Set to the code identifying the programming language of the calling program.
<i>comarealen</i>	Set to total number of two-byte words in <i>comarea</i> .
VPUTFIELD may set the following <i>comarea</i> items:	
<i>cstatus</i>	Set to nonzero value if call unsuccessful.
<i>numerrs</i>	May be decremented if new value is moved to a field which has error flag set.
<i>fieldnum</i>	Two-byte integer variable containing the number assigned to the field in the data buffer by FORMSPEC.
<i>fieldbuf</i>	Character string in an application containing the data to be copied to the specified data buffer field.
<i>buflen</i>	Two-byte integer containing the number of bytes to be copied from <i>fieldbuf</i> to the field identified by <i>fieldnum</i> .
<i>actualen</i>	Two-byte integer that specifies the number of bytes actually copied.
<i>nextfldnum</i>	Two-byte integer to which VPUTFIELD returns the number of the next field in screen order. If there are no more fields, it returns zero.

Discussion

The data in an application is copied to the field in the data buffer identified by its field number. Note that the field number is a unique number assigned to the field by FORMSPEC when the form is first created. The number assigned to a field by FORMSPEC does not change regardless of any changes to the field's position in the form or to its length. The field number can only be changed with the batch

command, **RENUMBER**, as described in Section 7. The field number must not be confused with the screen order number, which is the field's position in the data buffer and corresponds to its position in the form.

If the field is shorter than the data transferred to it, the data is truncated on the right. If the field is longer than the data transferred to it, the data, if any, in the remaining space in the field is not changed.

If the field whose data is being replaced contained an error, **VPUTFIELD** clears the field's error flag, and decrements *numerrs*.

Note that **VPUTFIELD** does not convert the data. To convert data from internal numeric representation to character strings, you must use **VPUTtype**, where *type* specifies the data type of the field in an application.

Example



```
COBOL
MOVE 1 TO FIELDNUM.
MOVE 10 TO FIELD-LEN.
MOVE "GADGET" TO PART-DES.
CALL "VPUTFIELD" USING COMAREA, FIELDNUM, PART-DES, FIELD-LEN,
DESC-LEN, NEXT-FIELD.
```

```
BASIC
250 F1=1
255 C=10
257 I$="GADGET"
260 CALL VPUTFIELD(C(*),F1,I$,C,A,N)
```

```
FORTRAN
FLDNUM=1
ICOUNT=10
XITEM="GADGET"
CALL VPUTFIELD(COMAREA,FLDNUM,XITEM,ICOUNT,INUM,FLDNXT)
```

```
SPL/PASCAL
INTEGER
  FLD'NUM,
  COUNT,
  ACTUAL'LEN,
  NXT'FLD'NUM;
BYTE ARRAY PART'DES(0:9):="GADGET ";
  FLD'NUM:=1;
  COUNT:=10;
  VPUTFIELD(COMAREA,FLD'NUM,PART'DES,COUNT,ACTUAL'LEN,NXT'FLD'NUM);
```

The calls shown above copy a 10-byte value from an application to field number 1 in the data buffer.

VPUTtype

Copies a numeric value of specified type from an application to a field in the data buffer in memory, converting the value to character set coded external representation.

Syntax

`VPUTtype {comarea,fieldnum,variable}[{,numdigits,decplaces}]*`

Parameters

type

The *type* in VPUTtype indicates that this intrinsic may be specified as:

VPUTINT	converts value to two-byte integer
VPUTDINT	converts value to four-byte integer
VPUTREAL	converts value to four-byte real value
VPUTLONG	converts value to eight-byte long value
*VPUTPACKED	converts packed decimal format to character set coded ASCII; this intrinsic has two extra parameters, <i>numdigits</i> and <i>decplaces</i> . Both are two-byte integer variables that contain the number of digits and number of decimal places, respectively, specified by the COBOL <i>usage is comp-3</i> data declaration.
*VPUTZONED	converts zoned decimal format to character set coded ASCII; has the parameters <i>numdigits</i> and <i>decplaces</i> , which are two-byte integer variables that contain the number of digits and number of decimal places, respectively, specified by the COBOL data declaration.
VPUTYYMMDD	converts six-byte character value to YMD, DMY or MYD

For example:

CALL "VPUTINT" USING COMAREA, FIELDNUM, VARIABLE

CALL "VPUTPACKED" USING
COMAREA, FIELDNUM, VARIABLE, NUMDIGITS, DECPLACES

comarea

Must be *comarea* name specified when the forms file was opened with VOPENFORMF.

VPUTtype

If not already set, the following *comarea* items must be set before calling VPUTtype:

cstatus Set to zero.

comarealen Set to total number of two-byte words in *comarea*.

VPUTtype may set the following *comarea* items:

cstatus Set to nonzero value if call unsuccessful.

numerrs May be decremented if new value replaces the value of a field with an error.

fieldnum Two-byte integer variable containing the field number assigned by FORMSPEC to the field in the data buffer to which the value is copied.

variable Variable of specified type in an application that contains the value to be converted to USASCII and copied to a field in the data buffer.

Discussion

Depending on how it is specified, this intrinsic converts integer, double, real, long, packed, zoned or yymmdd values to the external representation and copies the converted value to a particular field in the data buffer, right justified. (Note that the exact format depends on the data type of the destination field.) The destination field is identified by the field number assigned by FORMSPEC. The field to which the value is copied may be defined as a numeric field (data type NUM[n], IMPn, or DIG) or as a character field (data type CHAR).

NOTE

If you are using VPUTBUFFER in conjunction with an ARB, you do not need to use VPUTtype. VPUTBUFFER performs all the required conversions on the application data in the buffer.

You can use both VPUTBUFFER with an ARB and VPUTtype calls in the same program: the *buffercontrol* setting in the *comarea* that controls ARB processing can be switched on or off for each form.

Note that the field number never changes as long as the field exists. It is not changed when the position of the field in the form is changed, or its length or other characteristics are changed. The field number can only be changed with the batch command, RENUMBER, as described in Section 7. The field number should not be confused the screen order number, which is the position of the field in the data buffer and is based on the field position within the form. Thus, the field number provides a way to locate fields regardless of their position. If the specified field had an error, VPUTtype clears the field's error flag, and decrements *numerrs*.

Refer back to table 6-13 under VGETtype for the format of each of the data types that may be converted. Note that COBOL does not have type real or long, and BASIC does not have a double integer data type.

VPUTtype

Example

```
COBOL
77 FIELD-NUM PIC S9(4) COMP.
77 ITEM PIC S9(4) COMP.
  .
MOVE 4 TO FIELD-NUM.
MOVE 25 TO ITEM.
CALL "VPUTINT" USING COMAREA, FIELD-NUM, ITEM.
```

```
BASIC
260 F1=4
263 I=25
265 CALL VPUTINT(C(*),F1,I)
```

```
FORTRAN
FIELD=4
ITEM=25
CALL VPUTINT(COMAREA,FIELD,ITEM)
```

```
SPL/PASCAL
INTEGER FIELD,ITEM;
  .
FIELD:=4;
ITEM:=25;
VPUTINT(COMAREA,FIELD,ITEM);
```

The calls shown above convert an integer value of 25 in the application to the external representation and copy it to field 4 in the data buffer in memory.

VPUTWINDOW

Copies a message to the window area of memory.

Syntax

```
VPUTWINDOW {comarea,message,length}
```

Parameters

<i>comarea</i>	Must be <i>comarea</i> name specified when the forms file was opened with VOPENFORMF. If not already set, the following <i>comarea</i> items must be set before calling VPUTWINDOW:
<i>language</i>	Set to the code identifying the programming language of the calling program.
<i>comarealen</i>	Set to total number of two-byte words in <i>comarea</i> .
VPUTWINDOW may set the following <i>comarea</i> items:	
<i>cstatus</i>	Set to nonzero value if call unsuccessful.
<i>message</i>	Character string containing the message to be copied to the window area of memory.
<i>length</i>	Two-byte integer that specifies the number of bytes in the message. If set to zero, any message in the window is cleared to blanks. The maximum length is 150 bytes, but only 80 can be printable characters.

Discussion

This intrinsic copies the specified message to the window area of memory for later display. Then, a call to VSHOWFORM can be used to display the message in the window area of the terminal screen, with the window enhanced as indicated by *windowenh* of *comarea*. A message copied by VPUTWINDOW overwrites any previous message in the window area, including any message copied by a previous call to VPUTWINDOW or VSETERROR.

If the message is longer than the defined window length, the message is truncated on the right. If shorter, the rest of the window line is cleared.

Note that the forms file may be defined with no window line for error and status messages. In this case, the message is ignored.

FORMSPEC in batch mode allows you to manage a forms file without tying up your terminal. With this feature, forms may be deleted from or copied to a forms file, compiled, and listed on a printer by accepting commands from a job stream or from a disc file or I/O device.

Compiling Forms File in Batch Mode

Compiling a forms file can be very time-consuming, especially if there are many forms, fields, and field specifications defined in the file. By using FORMSPEC in batch mode, long compilations can be executed from a job stream. You do not have to wait for compilation to complete in order to use your terminal for other purposes.

Updating Forms in Batch Mode

A forms file can be updated systematically by using FORMSPEC in batch mode. Entire forms can be copied to or deleted from a forms file, field attributes may be updated, and function key labels may be created and updated. Individual screens, save fields, and some global information, however, cannot be defined in batch mode. Also, save fields cannot be copied from one forms file to another.

Listing Forms Files in Batch Mode

By using the batch mode LIST command, forms can be listed on the list device from a streamed job. This means forms file listings may be automated in a job that updates a forms file. For example, a job that updates a forms file could also print all or specific forms for future reference. The form listings may also be directed to a disc file to be examined.

Relating Forms in Batch Mode

Form family relationships can be established using the RELATE command in batch mode. The RELATE command requires that the forms exist and their screen designs be identical. Furthermore, the forms to be related must be distinct forms and the proposed child form cannot already be a member of a family of forms.

INVOKING FORMSPEC IN BATCH MODE

FORMSPEC in batch mode can be invoked from within a job stream or from within a session. When invoked, FORMSPEC can accept batch mode commands from a command file or from \$STDIN.

To invoke FORMSPEC to accept batch mode commands from a command file, use the MPE :RUN command with the INFO=*filename* parameter, where *filename* is the name of the command file. For example:

```
:RUN FORMSPEC.PUB.SYS;INFO="COMMANDS.PUB.MFG"
```

will execute FORMSPEC in batch mode using COMMANDS.PUB.MFG as the command file. Remember, before you can invoke FORMSPEC in batch mode to accept commands from a command file, you must first create the command file using a text editor and enter the batch mode commands into the file. (See the end of this section for a sample of a command file.)

The INFO=*filename* parameter directly invokes batch mode. In addition, FORMSPEC will always execute in batch mode within a job stream. When the INFO=*filename* parameter is omitted within a job stream, \$STDIN becomes the command file by default and commands are accepted from within the job stream. In order to use \$STDIN as the command file within a session, the INFO=*filename* parameter is required. For example:

```
:RUN FORMSPEC.PUB.SYS;INFO="$STDIN"
```

In response to the greater than (>) prompt, enter the batch mode commands. Although this usage of batch mode will not free up a terminal, users will have the ability to react to any error conditions which may occur.

When FORMSPEC detects an error in batch mode, the error message is sent to the output file (\$STDLIST by default). The number of errors can be checked with FORMSPECERRORJCW; number of warnings can be checked with FORMSPECWARNJCW. An error will cause one of the following actions:

- If any batch mode command except the FILE command causes an error, then FORMSPEC will skip that command and continue to the next command.
- If a FILE command caused the error or the error occurred because FORMSPEC did not recognize a batch mode command, then all batch mode commands, except the FORMS command, will be skipped until the next FILE or the EXIT command is encountered.

Batch mode output (errors, warnings, or messages) will normally be directed to \$STDLIST. However, output can be redirected to another file by using the MPE :FILE command to redefine the formal file designator, FORMOUT. For example, since \$STDLIST in a session is the terminal, you may want to redirect output to the line printer as follows:

```
:FILE FORMOUT;DEV=LP
```

When the output file is redirected to a disc file or the list device, the batch mode commands will also be echoed to the output file as they are executed. This will help to determine the cause of any errors that may occur.

BATCH MODE COMMANDS

The following pages show the syntax and description of the batch mode commands that FORMSPEC accepts in a command file or job stream. Table 7-1 gives a summary of the batch mode commands, a brief description of their functions, and the interactive counterpart for each command.

Note that each command must be entered on a separate line and a command may not carry over to the next line. Leading blanks may be used for indentation. You can also use blank lines between commands to make the command file easier to read. Comments may be inserted anywhere in a command file. They may be more than one line long, but they must be delimited by double arrows <<*comment*>>.

Table 7-1. FORMSPEC in Batch Mode Commands

Command	Function	Interactive Counterpart
COMPILE	Compiles the current forms file.	Selection (X) on the FORMSPEC Main Menu.
COPY	Copies forms within the current forms file, or from another file to the current forms file.	Selection (C) on the FORMSPEC Main Menu.
DELETE	Deletes a form from the current forms file.	Selection (D) on the FORMSPEC Main Menu.
EXIT	Terminates FORMSPEC in batch mode.	Function key F8 in any menu.
FIELD	Updates field attributes.	Field attributes specified on the FORMSPEC Field Menu.
FILE	Declares the current forms file.	Forms file name on the Forms File Menu.
FKLABELS	Creates or updates function key label strings.	FORMSPEC form-specific Function Key Menu.
FORMS	Prints summary information on the current forms file as well as for each or all the forms therein.	No interactive counterpart for this command.
LIST	Lists a form or all the forms on the standard list device.	Selection (L) on the FORMSPEC Main Menu.

Table 7-1. FORMSPEC in Batch Mode Commands, cont'd

Command	Function	Interactive Counterpart
RELATE	Establish family relation between two forms.	Selection (R) on the FORMSPEC Main Menu.
RENUMBER	Renumbers fields within a form.	No interactive counterpart for this command.
SELECTLANG	Updates native language specifications.	FORMSPEC Terminal/Language Menu.
SELECTTERM	Updates terminal and device specifications.	FORMSPEC Terminal/Language Menu.
ARB Commands	Function	Interactive Counterpart
ARBTOSCREEN	Set data type conversion defaults from ARB to screen.	Select defaults on FORMSPEC Data Type Conversions Menu.
SCREENTOARB	Sets data type conversion rules from screen to ARB.	Select defaults on FORMSPEC Data Type Conversions Menu.
DELARB	Deletes an ARB.	Select DEL on the FORMSPEC ARB Menu.
GENARB	Generates an ARB from a form.	Select GEN on the FORMSPEC ARB Menu.
ADDARBFIELD	Adds a field to the ARB.	Select ADD on the FORMSPEC Restructure ARB Menu.
DELARBFIELD	Deletes a field from the ARB.	Select DEL on the FORMSPEC Restructure ARB Menu.
MODARBFIELD	Modifies the attributes of an ARB.	Select MOD on the FORMSPEC ARB Menu.
MOVEARBFIELD	Moves a field or group of fields around on the ARB.	Select MOVE on the FORMSPEC Restructure ARB Menu.
RENAMEARBFIELD	Changes the name of an ARB field.	Select RENAME on the FORMSPEC Restructure ARB Menu.

DELETE

Removes a form from the current forms file.

Syntax

```
DELETE form
```

Parameters

form is the name of the form to be removed from the current forms file. If *form* does not exist, FORMSPEC will issue a warning and proceed to the next command.

Example

```
DELETE ORDER2
```

removes the form ORDER2 from the current forms file.

EXIT

Terminates FORMSPEC in batch mode.

Syntax

EXIT

Discussion

The EXIT command is optional. FORMSPEC will also terminate after the last command in the command file is executed.

Updates field attributes.

Syntax

```
FIELDform {fieldtag  
          {oldfieldname} } { newfieldname  
                           ,enhancement  
                           ,fieldtype  
                           ,fedatatype  
                           ,initialvalue }
```

Parameters

<i>form</i>	the name of an existing form within the current forms file.
<i>fieldtag</i>	the tag assigned to the field in form screen design.
<i>oldfieldname</i>	the name currently assigned to the field in the field menu.
<i>newfieldname</i>	replacement field name.
<i>enhancement</i>	replacement display enhancement code.
<i>fieldtype</i>	replacement field type.
<i>datatype</i>	replacement data type.
<i>initialvalue</i>	replacement initial value. Leading blanks are treated as part of the <i>initialvalue</i> . If \$EMPTY or \$empty is specified, the existing initial value is cleared.

Discussion

The first two parameters of this command identify the field to be updated. The remaining parameters are the actual updates. The field menu update parameters (*newfieldname* ... *initialvalue*) may be longer than the corresponding number of character positions available on the Field Menu in interactive FORMSPEC, in which case the input parameter is truncated.

EXAMPLE

```
FIELD PARTS a_b_c A_B_C_CODE,NONE,R,,D
```

Updates field menu selections for field *a_b_c* of form *PARTS*. The field name is set to *A_B_C_CODE*, the field display enhancement code is set to *NONE* (for no enhancements), the field type value is set to *R* (for required), and the data type value is unchanged. The initial value for the field is set to the character *D*.

FILE

Specifies the current forms file to be used.

Syntax

```
FILE file
```

Parameters

file is the name of the forms file to be used by subsequent batch mode commands. The *file* is referred to as the current forms file.

Discussion

The FILE command must precede all other commands in a command file or job stream. This command opens the specified forms file to be operated on by the other batch mode commands. Only one forms file can be modified at a time (only one current forms file at a time). However, more than one forms file can be operated on in one command file or job stream as long as a FILE command precedes any operations on that file. Note that a FILE command specifying a new current forms file closes the previous current forms file and opens the new current forms file. Termination of FORMSPEC automatically closes the current forms file.

If the file specified does not already exist, it is created. A message indicating that the file was created is issued.

Example

```
FILE INVFORMS.PUB
```

opens the forms file INVFORMS in the PUB group to be operated on by other batch mode commands.

Updates function key labels.

Syntax

```
FKLABELS form [\keylabel ... \keylabel]
```

Parameters

<i>form</i>	is the name of an existing form within the current forms file.
<i>keylabel</i>	literal which is to appear in the function key label. If \$EMPTY is entered as the first six characters of <i>keylabel</i> , the existing key label is cleared.

Discussion

The FKLABELS command creates (if none exist) or updates function key labels for the specified form. The first 16 characters of the *keylabel* literal are used for the label. Only the first eight *keylabels* are used.

Example

```
FKLABELS STAFF_SALARY \\ Start    Over\\$EMPTY\\ Exit  
Planner
```

Updates labels for function keys 2, 5 (which is cleared), and 8 in form STAFF_SALARY. Existing labels for function keys 1,3,4,6, and 7 are not updated.

FORMS

Prints a summary listing for the forms in the current forms file.

Syntax

```
FORMS [form]
      [@ ]
```

Parameters

- form* The name of an existing form within the current forms file for which summary information is to be printed. If omitted, a summary of the current forms file is printed.
- @ The commercial at sign (@) specifies that summary information for all forms, as well as the current forms file, is printed.

Discussion

The FORMS command prints the name of each form, the number of fields in the form, the number of bytes in the screen image, the number of children that the form has, and the name of the form's parent form, if any. If a form name (*form*) is included with the FORMS command, an extended summary is generated for the form. The extended summary includes all of the form information listed above. In addition, for each field in the form, the FORMSPEC assigned field number, data buffer offset (in bytes, one relative), and field length (in bytes) is printed. If the commercial at sign (@) is used for *form*, an extended summary is generated for each form in the forms file. See the examples below for samples of the three types of summary listings available.

Although all FORMSPEC output is normally directed to \$STDLIST, you may want to redirect this output separately to the line printer. To do so, use the MPE :FILE command to redefine the formal file designator, FORMOUT, as follows:

```
:FILE FORMOUT;DEV=LP
```

The FORMS command prints a maximum of 60 lines per page. This maximum can be changed by setting the JCW

```
FORMSPECLINESPERPAGE
```

See Appendix E for more information.

The FORMS command is a very useful batch mode command. When FORMS is used after the forms file has been updated, the listing can be used to quickly check that the resulting forms in the file are the correct ones.

This command also provides useful reference information when coding (or modifying) an application data buffer that has to match the layout of the fields in a form.

Lists a specified form or all forms on the list device.

Syntax

```
LIST [form]
```

Parameters

form is the name of the form in the current forms file to be listed. When *form* is not specified, all forms in the current forms file are listed.

Discussion

The LIST command can be used to provide documentation for a forms file. The forms are usually listed on the line printer. The LIST command can also be used to list forms to a disc file to be examined. To do so, use the MPE :FILE command to redefine the formal file designator FORMLIST as shown below:

```
:FILE FORMLIST=MYFILE;DEV=DISC;SAVE;NOCCTL
```

will list forms to the disc file, MYFILE. The SAVE keyword specifies that MYFILE be saved as a permanent file. The NOCCTL keyword indicates that no carriage control characters are written to the file. (See the *MPE Commands Reference Manual* (30000-90009), for more information on the MPE:FILE command.)

The LIST command prints a maximum of 60 lines per page. This maximum may be changed by setting the JCW

```
FORMSPECLINESPERPAGE
```

See Appendix E for more information.

Example

```
LIST
```

lists all forms in the current forms file on the list device.

```
LIST PARTS
```

lists the form PARTS in the current forms file on the list device.

RELATE

Creates a family relationship between two existing forms.

Syntax

```
RELATE childform TO parentform
```

Parameters

childform

is the name of the form which is to become the child form. It must already exist, and not be related to any other form.

parentform

is the name of the form which is to become the parent form. It must already exist, and have the identical form design as the *childform* specified with the *childform* parameter.

Example

```
RELATE FORM1 TO FORM2
```

Establishes a family relationship between FORM1 and FORM2 where FORM1 is the child form, and the FORM2 is the parent form.

Renumber fields to screen order.

Syntax

```
RENUMBER form
```

Parameters

form is the name of the form containing the fields to be renumbered.

Discussion

The fields in the form specified by *form* are renumbered in accordance with screen order, left to right, top to bottom. If your application depends on field numbers, be sure to update your application to reflect the changed field numbers. If the fields are already numbered in screen order, a message is returned and the fields are not renumbered.

For family forms, the parent form must be specified; all family members are then automatically renumbered. Again, if your application depends on field numbers, be sure to update your application, in this case, for all family members. If you specify the name of a child form, the fields are not renumbered. Instead, a message is returned stating the form was not renumbered because it is a child form and indicating the name of the parent form.

Example

```
RENUMBER FORM1
```

Renumbers the fields in FORM1 in screen order.

SELECTLANG

Updates native language specification.

Syntax

```
SELECTLANG [formsfilelangID][,FORMSPEClangID]
```

Parameters

formsfilelangID Set forms file native language specification.

FORMSPEClangID Set FORMSPEC native language specification.

Discussion

Only the first three characters of each parameter are used. Refer to "Setting Language ID Number" in Section 8 for more information. At least one parameter must be specified.

Example

```
SELECTLANG 8
```

Native language ID 8, German, is specified for the forms file.

```
SELECTLANG ,0
```

Native language ID 0, Native-3000, is specified for FORMSPEC. The native language specified for the forms file is retained.

Updates forms file terminal or device specifications.

Syntax

```
SELECTTERM [262xand239x][,color][,264x][,307x]
```

Parameters

<i>262xand239x</i>	Configure forms file for the HP 262X and HP'239X terminal families; also includes the HP 150.
<i>color</i>	Configure forms file for color terminals.
<i>264x</i>	Configure forms file for the HP 264X terminal family. Specifying yes (the value Y or y) enables function key labeling.
<i>307x</i>	Configure forms file for the data capture device family. Specifying eight (the value 8) enables support of the extended features of the HP 3081A.

Discussion

The parameters may be any valid non-blank character to select the terminal family. Only the first character of each parameter is used.

At least one paramter must be specified. Providing a USASCII blank as the specification for a given device/terminal family does not change the existing specification. Specifying \$EMPTY or \$empty clears an existing specification.

Example

```
SELECTTERM x,,y
```

Forms file is configured for HP 262X/239X terminals, the HP 150, and HP 264X terminals, with HP 264X function key labeling enabled. Existing color specification, if any, is retained.

```
SELECTTERM ,x,$EMPTY,8
```

Forms file is configured for color terminals and HP 307x terminals, with HP 3081A features enabled. HP 264X specification is cleared and existing HP 262X/239X/150 specification, if any, is retained.

ARB BATCH MODE COMMANDS

The following batch mode commands support the ARB feature. They are based on the ARB Menu options provided in interactive FORMSPEC. The commands are summarized below, and each one is described in detail in the rest of this section.

NOTE

In the ARB commands, *destination* is:

[BEFORE fieldname] or [AFTER fieldname]

The data type conversion commands are:

ARBTOSCREEN and ARBTOSCREEN

The ARB commands are:

DELARB and GENARB

The ARB field commands are:

ADDARBFIELD, DELARBFIELD, MODARBFIELD, MOVEARBFIELD, and RENAMEARBFIELD

Their functions are summarized in Table 7-1.

Add a field to the ARB.

Syntax



```
ADDARBFIELD arbname fieldname destination
```

Parameters

<i>arbname</i>	Name of the ARB to which the field will be added.
<i>fieldname</i>	The name of the new field.
<i>destination</i>	Position at which the field is to be inserted. May be left blank.

Discussion

If the *arbname* doesn't exist, an ARB with this name will be created for the associated form. The ARB cannot contain the *fieldname* already. If *fieldname* is not a field on the form, it becomes a filler field, with no datatype and with a length of 1 (this length may be changed using the MODARBFIELD command). If *fieldname* is on the form, its ARB *length* and *data type* are created from its screen attributes and the Data Conversion table (set up using the Data Conversion Menu). If *destination* is specified, the field will be inserted at that position, otherwise it will be appended to the end of the ARB.

ARBTOSCREEN

Create/update the datatype conversion rules for converting ARB field contents to screen field contents.

Syntax

ARBTOSCREEN [*chartype*, *inttype*, *realtype*, *packtype*, *zonetype*]

Parameters

<i>chartype</i>	The conversion type for ARB fields with type CHAR.
<i>inttype</i>	The conversion type for ARB fields with type INT or DINT.
<i>realtype</i>	The conversion type for ARB fields with type REAL or LONG.
<i>packtype</i>	The conversion type for ARB fields with type PACK, PACK0, PACK1, ... PACK9, SPACK, SPACK0, SPACK1, ... SPACK9.
<i>zonetype</i>	The conversion type for ARB fields with type ZONE, ZONE0, ZONE1, ... ZONE9.

Discussion

This command creates/updates the datatype conversion rules for converting ARB data types to screen data types. If a data type update is omitted, then the current default for that type is retained. If no update parameters are specified, all ARB-to-screen conversion types are set to CHAR, the default. Leading commas must be supplied.

Valid data type updates are: CHAR, YMD, MDY, DMY, DIG, NUM[n], NUMA, IMPn, IMPA

EXAMPLE

The command:

ARBTOSCREEN , DIG, NUM

sets the conversion type for INT and DINT ARB fields to DIG and for REAL and LONG ARB fields to NUM.

Deletes the entire specified ARB.

Syntax

```
DELARB arbname
```

Parameters

arbname Name of the existing ARB.

Discussion

All the fields are deleted. You can generate another ARB for the specified form if required. If extensive changes are made to a form, it is easier to delete the existing ARB and create a new one than to revise the old ARB.

DELARBFIELD

Deletes a specified field from the ARB.

Syntax

```
DELARBFIELD arbname fieldname
```

Parameters

arbname Name of the existing ARB.

fieldname Name of the specified field on the ARB.

Discussion

The *arbname* must exist in the forms file and the *fieldname* must exist on the ARB.

Generates an ARB from the form specified by the formname.

Syntax

GENARB *arbname*

Parameters

arbname Same as the name of the associated form.

Discussion

The ARB will contain a field for every field on the form. The data type and length of each ARB field are determined from the screen attributes and the Data Conversion table (set up using the Data Conversion Menu).

MODARBFIELD

Modify the attributes of a field on an ARB.

Syntax

```
MODARBFIELD arbname fieldname {[length, type]}
```

Parameters

<i>arbname</i>	Name of the ARB that contains the field to be modified.
<i>fieldname</i>	The name of the field to be modified.
[<i>length, type</i>]	The new ARB field length and/or data type.

Discussion

The *arbname* must exist in the forms file and the *fieldname* must exist on the ARB. You can change both the parameters *length* and *type*, but you must specify at least one. They must conform to the edit rules (see Section 3).

EXAMPLE

The command:

```
MODARBFIELD EMPLOYEE SSNUM , CHAR
```

changes the ARB data type to CHAR.

MOVEARBFIELD

Move a field or group of fields to a specified position on an ARB.

Syntax

`MOVEARBFIELD arbname range [destination]`

Parameters

<i>arbname</i>	The name of the ARB.
<i>range</i>	The field or fields to be moved, expressed as <code>fieldname</code> or <code>fieldname/fieldname</code>
<i>[destination]</i>	The position in the ARB to which the field(s) will be moved. Can be left blank.

Discussion

The *arbname* must exist. If the *destination* is not specified, the field(s) will be appended at the end of the ARB.

EXAMPLE

Suppose the fields in the ARB EMPLOYEE are in this order:

NAME
ADDRESS
PHONE
SSNUM
SUPERVISOR

The command:

`MOVEARBFIELD EMPLOYEE ADDRESS/PHONE AFTER SSNUM`

or the command:

`MOVEARBFIELD EMPLOYEE SSNUM BEFORE ADDRESS`

will rearrange the fields in this order:

NAME
SSNUM
ADDRESS
PHONE
SUPERVISOR

RENAMEARBFIELD

Change the name of *oldfieldname* to *newfieldname* in the ARB *arbname*.

Syntax

```
RENAMEARBFIELD arbname oldfieldname newfieldname
```

Parameters

<i>arbname</i>	Name of the ARB that contains the field.
<i>oldfieldname</i>	The current name of the field.
<i>newfieldname</i>	The new name you want to give the field.

Discussion

This command does not change the ARB data type or length. The *arbname* must exist in the forms file, and the *oldfieldname* must exist on the ARB. The ARB cannot already contain a field with the *newfieldname*. If *newfieldname* is not a field on the form, the renamed field becomes a filler field.

SCREENTOARB

Create/update the datatype conversion rules for generating an ARB from a form.

Syntax

SCREENTOARB [*chartype*, *datetype*, *digtype*, *numtype*, *imptype*]

Parameters

<i>chartype</i>	The conversion type for screen fields with type CHAR.
<i>datetype</i>	The conversion type for screen fields with a date type.
<i>digtype</i>	The conversion type for screen fields with type DIG.
<i>numtype</i>	The conversion type for screen fields with type NUM, NUM0, NUM1, ... NUM9.
<i>imptype</i>	The conversion type for screen fields with type IMP, IMP0, IMP1, IMP9.

Discussion

This command creates/updates the datatype conversion rules for generating an ARB from a form. If an update parameter is omitted, then the current conversion type for that parameter is retained. If no update parameters are specified, all screen-to-ARB conversion types are set to CHAR, the default. Leading commas must be supplied.

Valid type updates are: CHAR, YYMMDD, INT, DINT, REAL, LONG, PACKn, SPACKn, ZONE_n, PACKA, SPACKA, ZONEA

EXAMPLE

The command:

```
SCREENTOARB ,INT, REAL, REAL
```

sets the conversion type for DIG screen fields to INT, and for NUM and IMP screen fields to REAL.

VPLUS/V Error Messages

828	Illegal combination of options for the Multifunction reader.	The combinations HOLES with Clock On and MARKS with NONE are not valid.
829	Invalid Multifunction reader definition (COD, CAD, NONE).	Enter one of the valid Multifunction reader definitions.
830	Invalid barcode type (UPC, EAN, I25, I39, MAT, ILV).	Enter one of the valid Barcode Reader types.
831	Internal error: Writing device intermediate record (FSERR !!!!)	
832	Invalid combination of PAUSE INTERVAL and WAIT FOR USER	If Split Message Pause is specified, Wait for User must be NO.
833	Enter "CONTINUE" only, or press a function key	Only CONTINUE may be entered in this field.
834	NEXT is not appropriate until the forms file has been adjusted	The specified forms file is an old version and must be adjusted before going further.
835	Place name of form to be listed within List Form field.	
836	Enter "Y" or "N" only.	
837	Enter "Y" to go to Data Type Conversion menu.	

Init and Compile Errors

Error	Message	Cause/Action
902	For form !, the next form ! has not been defined.	The next form name specified does not exist in the forms file. Change the next form name or create a form with that name.
903	The head form ! has not been defined.	The head form specified on the globals menu does not exist in the forms file. Create a form with the head form name or change the head form on the globals menu.
904	Internal error closing FORMSPEC forms file. (COM'STATUS !!!!)	

905 Internal error closing
the terminal file.
(COM'STATUS !!!!)

VPLUS/V Error Messages

1932	Internal error: Operand must be character string constant.
1933	Internal error: Invalid operand in APPLY.
1934	Internal error: Invalid operand in type check.
1935	Internal error: Invalid operand in MATCH.
1936	Internal error: Invalid operand in CDIGIT.
1937	Internal error: Invalid right operand.
1938	Internal error: Invalid left operand.
1939	Internal error: Invalid source operand.
1940	Internal error: Invalid Dtype in type checking.
1941	Internal error: EXTIN conversion failed.
1942	Internal error: Bad id number.
1943	Internal error: Bad screen order number.
1944	Internal error: MEM dispose failed.
1945	Internal error: Lexic level underflow.
1946	Internal error: Invalid leading blanks.
1947	Internal error: Invalid Dtype in scanner.
1948	Internal error: MEM allocate failed.
1949	Internal error: Invalid production.

1950 Internal error: Scanner error.

1951 Invalid local edit combination. An invalid combination of local edits has been specified in the CONFIG phase of the field processing specifications.

Pattern Compile Errors

Error	Message	Cause/Action
2101	A "(", ")", "[", or "]" is not paired in pattern.	MATCH statement contains odd number of parentheses or brackets.
2102	Invalid character found in pattern.	A character in the expression is not recognized as a valid character in the pattern.
2104	End of pattern found prematurely.	Pattern characters were found after the end of the pattern was found.
2105	Second character in range is lower than first.	A range (a:b) is transposed, or else a colon was not preceded by an exclamation point.
2203	Pattern is too big.	Either split the pattern into smaller patterns (with IF statements), or simplify the pattern.
2900	Internal error: COMPILE'PATTERN failed.	

FORMSPEC B.05.xx Release Errors

Error	Message	Action
3097	Invalid data type conversion notation.	Enter the correct data type to which the specified data type should be converted.
3098	Data Type Conversion Record has not been defined.	Use the Data Type Conversions Menu to set the default data type conversions.

List Process Errors

Error	Message	Action
3100	Internal error: Could not create list process.	
3101	Internal error: Could not send mail to list process.	
3102	Internal error: Could not activate list process.	
3103	Internal error: Mail from list was lost.	
3104	Formsfile reopen failed. Press PREV to reopen. (FSERR !!!!)	
3105	List could not open the forms file. (FSERR !!!!)	
3106	Internal error : List could not close the formsfile.	
3107	Internal error: could not flush current form to the forms file.	
3108	Internal error: could not flush the field table to the forms file.	
3109	Internal error: could not flush the globals to the forms file.	
3110	Internal error: could not close the formsfile.	
3111	List could not open the FORMSPEC formsfile. (FSERR !!!!)	
3112	Internal error: List could not close the FORMSPEC formsfile.	
3113	List could not find form ! in the FORMSPEC formsfile.	
3114	List write failed. (FSERR !!!!)	

- 3115 FORMSPECLINESPERPAGE JCW must
be between 1 and 150.
List failed.
- 3116 FORMSPECLINESPERPAGE JCW must
be between 1 and 150.

Application-ready Buffer (ARB) Errors

Error	Message	Cause/Action
3403	Internal error: ARB detail record missing.	
3405	This field must not be blank.	Enter the required information in the field.
3406	This field accepts only contiguous digits.	Remove any spaces between the digits.
3407	Must specify number of decimals here.	
3409	Internal error: could not update ARB Detail record.	
3412	Internal error: Field Symbol Table record missing.	
3413	The form has no fields.	You cannot reproduce the form fields on the ARB because they do not exist.
3414	Internal error: Data Type Conversion record missing.	
3416	Internal error: could not create ARB Detail record.	
3418	Internal error: could not create ARB Map record.	
3420	An ARB already exists.	You cannot create two ARBs for one form. If you want to create a new ARB, you must delete the old one first.

3423	Warning: run-time errors may result from default data conversions.	You may have specified incompatible data conversions. Check your data type conversion record to make sure the data conversions are legitimate. If your code has made provision for these anomalies, you may disregard the warning.
3424	Warning: ARB length adjusted to match data type.	You specified a length for an ARB field that was too long or too short.
3425	Length must be greater than zero.	
3426	Field was not found in ARB.	
3429	ARB does not exist.	
3432	Selection must be one of the abbreviations listed below.	
3450	First field of range is missing; specify a name or number.	
3451	Number specified is not in current ARB range.	
3452	ARB is full; no more fields may be added.	An ARB may hold up to 256 fields.
3453	Please enter a valid ARB field name.	Check the syntax of the name you have entered.
3455	Range is reversed; try 'leading-field/trailing-field'	You have reversed the order of the fields you want to manipulate. The field that appears first on the ARB must appear first in the command.
3456	Please enter a valid command (Add, Move, Rename or DELETE).	
3457	That command cannot be abbreviated. Please try again.	
3458	ARB field name is too long; maximum length is 15 characters.	

VPLUS/V Error Messages

- | | | |
|------|---|--|
| 3459 | Please specify Before or After, or leave blank. | |
| 3460 | ARB field name to be added already exists in ARB. | You cannot duplicate an ARB field name; choose a new name or delete the other field first. |
| 3461 | Internal error: Selection routine approved an unknown command. | |
| 3462 | New {Destination} name already exists in ARB. | Select a name that does not already exist. |
| 3465 | A destination entry requires that Before or After be specified. | If you do not want to specify Before or After, you must leave the Destination field blank. |
| 3466 | An ARB field name must start with an alphabetic character. | |
| 3467 | No imbedded blanks are allowed in an ARB field name. | |
| 3468 | Legal ARB field name characters are A-Z, 0-9, and ' '. | |
| 3469 | An ARB field name cannot be a reserved word. | Choose a new name for the field. |
| 3473 | ARB field name must be specified; numeric reference is not allowed. | |
| 3474 | Second field of range is missing; specify ARB field name or number. | |
| 3475 | Internal error: SearchRec did not locate field in ARB Add | |
| 3476 | A range cannot be moved to a destination within itself. | |
| 3498 | Internal error: VGETBUFFER failed (COM'STATUS !!!!) | |

3499 Internal error: VPUTBUFFER
failed. (COM'STATUS !!!!!)

NOTE

In most cases, the corrective action is clearly implied by the message.
"Internal Errors" are serious; they imply a corrupted forms file and/or a
program error. Call your SE.

ARB Compile Errors

Error	Message	Action
3500	In ARB !, field ! has no corresponding screen field.	Either delete the ARB field, or add a field to the associated screen.
3501	In ARB !, field ! has an incorrect length.	Change the ARB field length to match the length of the corresponding screen field.
3502	In ARB !, field ! has an incorrect ARB type.	Check the permitted conversion types and correct the ARB data type.
3503	Internal error: In ARB !, missing source record for field !.	

More Menu Init and Processing Errors

Error	Message	Cause/Action
7001	Correct syntax is: "FIELD formname fieldtag/name" then menu specs.	
7002	Unable to open screen file.	
7003	WARNING: No new screenfile specified. Old screenfile was closed.	
7004	Correct syntax is: "ADD formname [\comments]".	
7005	Correct syntax is: "READD formname [newformname] [\comments]".	
7006	A screen layout file is not currently open.	
7007	End of file when reading screen layout file.	
7008	File system error when reading screen layout file.	
7009	Form name in screen file record record not equal to current form name.	
7010	Form contains no fields.	
7011	Cannot find screen record for form.	
7012	Write of screen record to screen layout file failed.	
7013	WARNING: if ADD command, form created with no screen	
7014	WARNING: if READD, screen not updated (form renamed?).	
7015	Correct syntax is: "FKLABELS formname [\keystring1...\keystringn]".	
7016	Screenfile contains an invalid character.	
7017	Missing datatype parameter.	

- 7018 Missing field type parameter.
- 7019 Missing field enhancement parameter.
- 7020 SELECTTERM update parameters must be separated by commas.
- 7021 Missing new FORMSPEC language ID.
- 7022 FIELD update parameters must be separated by commas.
- 7023 Renumbered !.
- 7024 Renumber was not necessary for !.
- 7025 Form is family child; renumber family by specifying parent form !.
- 7026 Internal error: unable to fetch form head.
- 7027 There are no forms in the current file.
- 7028 Creating new forms file.
- 7029 Recovering file -- PLEASE WAIT
- 7030 Compiling form !
- 7031 Number of errors = !!!!
Number of warnings = !!!!
- 7032 There are no compiled forms in the current file.
- 7033 No update values specified.
Command ignored.
- 7034 **** End of compiled forms listing. ****
- 7035 Missing color parameter.
- 7036 Missing 264x parameter.
- 7037 Missing 307x parameter.
- 7038 New function key labels must be preceded by \.



VPLUS/V Error Messages

7039 End of command file.

7040 Forms file is being converted to new format.

7041 Missing ARB name.

7042 Sorry. That command has not been implemented.

7043 Missing ARB field name.

7044 Correct syntax is:

7045 ADDARBFIELD ARBname fieldname
[BEFORE/AFTER fieldname]

7046 Missing field name in
"BEFORE/AFTER fieldname".

7047 MODARBFIELD ARBname fieldname { length, type }

7048 MOVEARBFIELD ARBname field [/ field] [BEFORE/AFTER field]

7049 SCREENTOARB [char, date, dig, num, imp]

7050 ARBTOSCREEN [char, int, real, pack, zone]

7051 RENAMEARBFIELD ARBname oldfieldname newfieldname

Native Language Support Errors

Error	Message	Cause/Action
8000	This edit statement is not valid for international forms files.	
8001	Edit contains data inconsistent with language of this forms file.	
8002	Edit contains language-dependent data.	
8003	Edit contains data not compatible with the language of this forms file.	
8004	Invalid initial value for international forms file.	
9001	Native language support software not installed.	

- 9002 Language specified is not
 configured on this system.
- 9003 Character set specified is
 not configured on this system.
- 9004 Internal error: National table
 is not present.
- 9005 Internal error: Bad NLT extra
 data segment.
- 9006 Internal error: Bad LDST extra
 data segment.
- 9007 Error calling native language
 support intrinsic.
- 9008 Error calling native language
 support intrinsic.
- 9009 Error calling native language
 support intrinsic.
- 9010 Internal error: NLINFO item
 out of range.
- 9011 WARNING: Language not configured,
 change or hit ENTER to proceed.
- 9012 This edit requires a configured
 language.
- 9014 Attempted setting a language-
 dependent forms file to another
 language.
- 9015 NATIVE-3000 is currently the
 only selection available.
- 9016 Invalid date value for
 international forms file.
- 9017 Internal Error: Error updating
 language ID - try again.
- 9070 Internal Error: writing terminal
 selection source record failed.
- 9500 Language of forms file is not
 configured on this system.

VPLUS/V Error Messages

- 9998 Language ID must be 0 to 999
 or -1 for international forms
 file.
- 9999 Call to native language support
 intrinsic failed.
-

012	The HP2640B needs straps A, C, H in (closed); D, F, G out.	You must strap this terminal (and the 2644A) yourself; other terminals are strapped automatically (see App. G).
013	The HP2644A needs straps A, C, H in (closed); D, G out.	You must strap this terminal (and the 2640B) yourself; other terminals are strapped automatically (see App. G).
014	The HP2645A/41A/48A needs straps A, C, H, J in (closed); D, G out.	Note that these terminals should be configured automatically by VPLUS.
015	Terminal type incorrectly determined.	Possible block mode or data comm problem. Check straps. Note: 2645 multipoint straps should never be touched!
016	Internal error: HP2645A multipoint terminal mode not correctly set.	See error 015.
017	Comarea too short for run-time HP3075/6 terminals.	The Comarea must be 85 words for HP 3075/6 terminals. Set Comarealen to 85.
018	Non-page/block mode terminal types not supported.	Redefine terminal type. (Note: VPLUS requires a terminal which supports page/block mode.)
019	Internal error: FDEVICECONTROL should not return ccg.	See System Manager.
021	Invalid terminal type.	TERMTYPE used is a printer terminal type.

----- General Intrinsic Error Messages

Error	Message	Cause/Action
020	Error detected by host language.	Check the Comarea item FILERRNUM for the host language error number.
021	Invalid terminal type.	See 021 above.

VOPENTERM Error Messages (continued)

Error	Message	Cause/Action
023	Environment control file access failed	

VOPENFORMF and VCLOSEFORMF Messages

Error	Message	Cause/Action
040	Internal error: DLSIZE failed; there is not enough DL area space.	Stack size or MAXDATA too small. Re-PREP your application program with a larger MAXDATA parameter.
041	Internal error: DLSIZE failed; the stack is frozen.	Stack size or MAXDATA too small. Re-PREP your application program
042	Formsfile name not passed to VOPENFORMF.	The name of the forms file to be opened must be included in the call to VOPENFORMF.
043	Comarea length may not be less than 60.	The Comarea length as specified by the item Comarealen must be 60 words for HP terminals and 85 words for data capture devices.
044	Comarea length is too large.	The Comarea length as specified by the item comarealeng must not be longer than the longest length documented.
045	TERMINAL or FORMSFILE has not yet been opened.	
046	Unrecognized Comarea language code passed.	
050	Forms File open failed. (FSERR !)	FOPEN intrinsic failed. Check forms file name.
051	The file is not a VPLUS Forms File.	Check the formfile parameter to make sure the file name is correct.
052	Internal error: Forms File FGETINFO failed. (FSERR !)	FGETINFO intrinsic failed.

- | | | |
|-----|--|--|
| 053 | Forms File probably
hasn't been compiled.
(FSERR !) | Run FORMSPEC and compile the
forms file. |
| 060 | The program supplied
COMAREA extension is too
small. | If you are using BASIC, check
word 4 of COMAREA (USRBUFLN),
and increase the length of the
COMAREA extension. |
| 061 | Internal error: Failure
to obtain required PASCAL
Heap area. | |

VPLUS/V Error Messages

351	Overflow in a negate operation.	See error 345.
360	Attempted a SET from a field containing an invalid date.	A date is expected as the source of a SET statement, but the date is not valid.
363	Internal error: Invalid op in double eval.	
364	Index expression out of range in an "index of" date statement.	See error 350. (Date operands)
370	Internal error: Invalid op in text eval.	
371	Index expression out of range in an "index of" text statement.	See error 350. (CHAR operands)
380	The index expression evaluates to "empty."	An arithmetic expression used as the index in an index OF element, element.. operand is blank or zero.

----- VREADBATCH Messages

Error	Message	Cause/Action
400	Can't read the batch file record. (FSERR !)	FREADDIR intrinsic failed.
401	Warning: Can't browse a batch file with variable length records.	Browse mode is illegal for batch records which are not fixed length.

----- VWRITEBATCH Messages

Error	Message	Cause/Action
450	Can't write the batch record. (FSERR !)	FWRITEDIR intrinsic failed.

----- VSETERROR, VGETFIELD, VPUTFIELD, VPUTtype, and VGETtype Messages

Error	Message	Cause/Action
500	A field with the field number supplied does not exist.	The fieldnum parameter contains a value that is not a field number in the current form.

VPLUS/V Error Messages

501	The field number supplied is out of range.	The fieldnum parameter contains a value greater than any field number associated with the current form.
502	The field requested is in error.	The field specified by the fieldnum parameter contains invalid data.
503	The field requested is empty.	The field specified by the fieldnum parameter contains no data.
504	Error converting field to numeric type.	VGETtype tried to convert value to a numeric type, and failed. Check conversion rules. (For VGETINT, message indicates attempt to convert number > 32767; receiving field is unchanged.)
505	Error converting numeric type to ASCII.	VPUTtype tried to convert a numeric value to a character string and failed. Check conversion rules.
506	A DIGIT field cannot contain a negative number.	A value in a DIG type field has a minus sign; DIG values may only be digits (0-9).
507	Function key labels option is disabled.	The COMAREA item LABEL'OPTION must be set to one before opening the forms file.
508	Error reading function key values from forms file (FSERR !!!!!)	An error occurred while trying to read the function key label definitions from the forms file.
509	Invalid input parameter to function key intrinsics.	An invalid value has been passed as the FORM-OR-GLOB or KEY-NUM parameter to VGETKEYLABELS, VSETKEYLABEL, or VSETKEYLABELS intrinsics.
510	A field with the screen order supplied does not exist.	

VGET/PUTBUFFER ARB Messages

Error	Message	Cause/Action
519	ARB flag is set but no ARB record is found.	Application-specified conversion but the forms file does not contain an ARB record. • Initialize ARB conversion flag to 0 if conversion is not desired. • Use FORMSPEC to add the necessary ARB record. • Check for possible corruption of the formsfile.
520	ARB conversion type specified is not supported.	Check for possible corruption of the formsfile.
521	ARB record length exceeded maximum limit.	Check for possible corruption of the formsfile.
522	IEEE conversion is not supported in this version.	Application-specified IEEE conversion which is not yet supported. Check the IEEE conversion flag in the Comarea.
523	ARB record does not contain the minimum amount of information required.	Check for possible corruption of the formsfile.

VGETLANG and VSETLANG Messages

Error	Message	Cause/Action
9002	Language specified is not configured on this system.	Select another language or check with System Manager to configure the desired language.

9014	Attempted setting a language dependent forms file to another language.	VSETLANG can only be used with international forms files.
------	--	---

VCHANGEFIELD

Error	Message	Cause/Action
850	Invalid field type specification.	Field type incorrect; use one of the listed codes.
851	Invalid data type specification.	Data type incorrect; use one of the listed codes - n indicates number of decimal positions.
852	Invalid enhance type specification.	Enhancement code is incorrect; check and correct.
853	Internal error: invalid field type.	Internal.
854	Internal error: invalid data type.	Internal.
855	Internal error: invalid enhance type.	Internal.
856	Invalid VCHANGEFIELD specification type.	Value must be 1-6.

INFO Intrinsic Messages

Error	Message	Cause/Action
601	Invalid input parameters to INFO intrinsic.	
602	Form specified to FORMINFO does not exist.	
603	INFO intrinsic: Form field list read error (FSERR !!!!!).	
604	INFO intrinsic: Form record read error (FSERR !!!!!).	
605	FF version is old. Must recompile to use FORMINFO intrinsic.	

606	Invalid field specified to FIELDINFO intrinsic.	Make sure that the field name is correctly spelled; that the field name parameter is 16 bytes long and padded with ASCII blanks. Make sure the field number is the actual field number.
607	Invalid form # specified to FORMINFO intrinsic.	

Forms Loading Errors

Error	Message	Cause/Action
700	No such form in terminal form storage memory.	The specified form is not purged because it does not exist.
701	Terminal form storage memory is full.	The specified form is not loaded because the form storage memory is full.
702	Requested load of blank form name.	

Reformat Messages

Error	Message	Cause/Action
2	Error opening reformat file. (FSERR !!!!!)	Check file system error number.
4	Batch file could not be opened. (FSERR !!!!!)	Check file system error number.
5	No reformatting was done.	No sequence of forms in batch matched with any Reformat sequence.
6	Batch file: !	Not an error. Listing message.
8	Error closing batch file. (FSERR !!!!!)	Check file system error number.
9	Error closing output file. (FSERR !!!!!)	
10	Error closing reformat file. (FSERR !!!!!)	
11	Error opening output file. (FSERR !!!!!)	

9004	Internal error: National table is not present.	Internal error from NLS.
9005	Internal error: Bad NLT extra data segment.	Internal error from NLS.
9006	Internal error: Bad LDST extra data segment.	Internal error from NLS.
9007	Error calling native language support intrinsic.	Internal error from NLS.
9008	Error calling native language support intrinsic.	Internal error from NLS.
9009	Error calling native language support intrinsic.	Internal error from NLS.
9010	Internal error: NLINFO item out of range.	Internal error from NLS.
9011	WARNING: Language not configured, change or hit ENTER to proceed.	Language specified is not configured on the system, forms file produced will only run on a system configured with that language.
9012	This edit requires a configured language.	
9013	Encountered invalid Asian (2-byte) character.	
9014	Attempted setting a language dependent forms file to another language.	VSETLANG can only be used with international forms files.
9015	NATIVE-3000 is currently the only selection available.	FORMSPEC language can only be 0 in this version.
9016	Invalid date value for international forms file.	
9017	Internal error: Error updating language ID - try again.	

VPLUS/V Error Messages

- | | | |
|------|--|--|
| 9070 | Internal error: writing terminal selection source record failed. | |
| 9500 | Language of forms file is not configured on this system. | |
| 9998 | Language ID must be 0 to 999 or -1 for international forms file. | Forms file language ID must be between -1 and 999. |
| 9999 | Call to native language support intrinsic failed. | |

VPLACECURSOR intrinsic

- | | | |
|-----|---|--|
| 800 | Cannot place cursor to display only fields. | Field number must be for an input field. |
|-----|---|--|

Native Mode Access Error Messages

- | | | |
|-----|---|-----------------|
| 998 | Negative argument passed to VPLUS intrinsic. | See Appendix J. |
| 999 | Attempt to access Compatibility Mode VPLUS services failed. | See Appendix J. |
-

The second file equation which VPLUS/V searches for is FORMLIST. The VPRINTFORM intrinsic uses FORMLIST to list forms (as in ENTRY). FORMLIST defaults to the system printer, but the user can save the list file on disc using the following file equation, which creates a file named FORMLIST containing the form listing.

:FILE FORMLIST; DEV=DISC;SAVE

If your system has both a page printer and a line printer (default), the file equation:

:FILE FORMLIST;DEV=PP

sends the form listing to the page printer.

The third file equation which VPLUS/V uses deals with a summary listing of the forms file from batch mode FORMSPEC. Batch mode prints to a file named FORMOUT. A file equation for FORMOUT can use the same parameters as discussed for the file FORMLIST.

10. The LIST command and the batch mode FORMS command print a maximum of 60 lines per page. This maximum may be altered by setting the JCW

FORMSPECLINESPERPAGE

For example, the MPE command

:SETJCW FORMSPECLINESPERPAGE=45

sets the maximum number of lines per page to 45. This JCW is useful when more white space is desired at the bottom of the page, or when special paper is used.

FORMSPECLINESPERPAGE must be in the range 1 to 150. Any value outside this range will cause a LIST command to fail, and will cause batch mode FORMSPEC to halt with the error message

FORMSPECLINESPERPAGE must be between 1 and 150

No lines will be printed in either case, to ensure that FORMSPEC does not waste paper printing a listing or FORMS reference in an unwanted format.

RULES FOR THE DL AREA

VPLUS/V makes use of the DL area for buffers and internal tables. Therefore, if the DL area is put to any other use during the execution of a VPLUS/V application, such use must not conflict with that of VPLUS/V. To insure against conflict, observe the following rules:

1. The first use of the DL area by an application, including any procedures, routines, or intrinsics executed on behalf of the application, should be the call to VOPENFORMF that opens the forms file and allocates the DL area needed by VPLUS/V.
2. The call to VCLOSEFORMF that frees up the DL area used by VPLUS/V must not occur until the application has released all other DL space allocated subsequent to the VOPENFORMF call.

These restrictions need not be followed if the application is wholly coded in any combination of Pascal, HPFORTRAN 77, and HPBUSINESS BASIC *and* sets the *comarea* item *language* to 5, which causes VPLUS/V to use the heap procedures provided by Pascal instead of managing the DL area directly.

VPLUS/V applications coded in other languages need to take great care if they pass the *language* identifier of 5 as a means of getting around the restrictions stated above. Specifying an incorrect *language* identifier can cause other aspects of the VPLUS/V interface, such as addressing of byte parameters, to function unreliably. In such a case, the application must use the INTRINSIC mechanism (CALL INTRINSIC in COBOL) for the main and all interacting parts of the application. Check the Pascal and COBOL reference manuals for more information.

In all cases, check for any other use of the DL area by intrinsics such as the DSG intrinsics.

This appendix contains information about terminals that affects the operation of VPLUS/V. An overview of the supported terminals and their main features is followed by information on terminals, grouped according to terminal family and model within the family. This appendix also contains information about:

- The VPLUS/V *comarea* items that relate to terminals
- Configuring terminal buffers
- Recovering from unexpected program interruption
- The user environment control file
- Advanced terminal I/O procedures; VTURNON/VTURNOFF

SUPPORTED TERMINALS AND FEATURES

VPLUS/V can be used with the following terminals:

HP 150	HP 2392A	HP 2622A	HP 2640B	HP 3075A
	HP 2393A	HP 2623A	HP 2641A	HP 3076A
HP 2382A	HP 2394A	HP 2624A	HP 2642A	HP 3081A
	HP 2397A	HP 2624B	HP 2644A	
		HP 2625A	HP 2645A	
		HP 2626A	HP 2647F	
		HP 2627A	HP 2648A	
		HP 2628A		

Applicable special features of these terminals are shown in Table G-1 and are described below.

Termtypes

As a user of an HP 3000 system, you may choose to specify a *termtypes* when you log onto MPE; for example,

```
:HELLO username.acctname;TERM=nn
```

However, VPLUS/V only supports a subset of the available *termtypes*. They are *termtypes* 10, 12, 14, and 24.

A *termtypes* that is not supported by VPLUS/V may be:

- Rejected as an error if VPLUS/V knows that it cannot continue, or
- Mapped to a supported *termtypes* if this will allow VPLUS/V to continue.

When VPLUS/V maps an unsupported *termtypes* to one that is supported, the terminal driver is reconfigured to the original *termtypes* when VPLUS/V closes the terminal. Finally, VPLUS/V defers to the Workstation Configurator when editing *termtypes* if that facility is available.

Modified Data Tag

You do not have to take any action to utilize this feature. Only the fields which have been modified are transmitted to the computer. This feature is supported on the following terminals:

HP 150	HP 2393A	HP 2624A/B
	HP 2394A	HP 2625A
	HP 2397A	HP 2628A



Extended Local Edits

This feature provides for the editing of information by the terminal as it is typed into the form. You specify the editing to be done by using the command **LOCALEDITS** in the field processing specification section. The HP 2624B and HP 2394A terminals support this feature.

Relabeling Function Keys

This feature allows you to specify during forms creation or modification time function key labels that will be used during data entry. Function key labeling is supported by **VPLUS/V** on all terminals except the data capture devices. On HP 264X terminals, **VPLUS/V** uses the last two screen lines to emulate function key labels, which are available as a terminal feature on all other HP terminals.

Security Display Enhancement

This feature allows you to inhibit the display of the data in certain fields. When this feature is used, the characters entered at the terminal are displayed as blanks on the screen. The security display enhancement is supported on the following terminals:

HP 150	HP 2393A	HP 2624B
	HP 2394A	HP 2625A
		HP 2626A
		HP 2628A

Terminal Information

USER ENVIRONMENT CONTROL FILE

The user environment control file is called VENVCNTL.PUB.SYS, and is designed to let the programmer assert limited control over terminal activities. It is opened and read on the VOPENTERM call, and it controls these two items in the user environment.

1. Abbreviating a terminal query. When VOPENTERM is called, it usually goes through a complicated process to identify the terminal type. The VENVCNTL file shortens the identification process. However, it should only be used if it is known that a query will elicit a valid terminal ID. To shorten the terminal query, enter 1 in the first column of the VENVCNTL file.
2. Enabling ☐AIDS/☐MODES/☐USERKEYS without a program. Instead of setting the SHOWCONTROL word of the user's *comarea* in the application, enter 1 in the second column of the VENVCNTL file. This option is very useful if you do not have the source code for the software but you want to enable ☐AIDS/☐MODES/☐USERKEYS.

To implement a user environment control file, build this file:

```
:BUILD VENVCNTL.PUB.SYS;DEV=DISC;REC=-80,1,F,ASCII
```

Use any editor to enter 1 in column 1 and/or 2. When VOPENTERM executes, it looks for the VENVCNTL file. If it does not find it, VPLUS/V carries out its usual operations. If the file exists and columns 1 and/or 2 are set, VPLUS/V performs either or both of the prescribed actions.

The VENVCNTL file need not necessarily be in PUB.SYS. However, if you have the file residing in another group and account, you must set up the file equation:

```
:FILE VENVCNTL.PUB.SYS=VENVCNTL.mygroup.myaccount
```

ADVANCED TERMINAL I/O PROCEDURES

In general, the VPLUS/V terminal input/output procedures such as VOPENTERM, VCLOSETERM, VREADFIELDS and VSHOWFORM are sufficient for most applications that communicate with a user via a terminal interface.

In building complex user interfaces, however, application designers have a limited number of tools for implementation. Hence, they resort to using FREAD, FWRITE, and input/output verbs of their programming language to display and retrieve information from the terminal when running VPLUS/V. The situation becomes more complicated when the application controls the terminal settings as well as the driver settings, and is likely to result in an application that runs in a restricted environment; for example, on an ATP and on one kind of terminal only.

VPLUS/V provides two additional procedures that help application designers to write applications with complex user interfaces that are more portable and easier to maintain. They are VTURNON and VTURNOFF.

VTURNON/VTURNOFF

VOPENTERM configures the driver and the terminal for block mode access. VCLOSETERM configures the terminal and driver for character mode access. These procedures initialize the terminal in many ways, including clearing the screen image. VTURNON and VTURNOFF reconfigure the terminal and driver but leave the terminal screen image intact. VTURNOFF allows the application to switch momentarily to character mode from block mode without disturbing the screen. A call to VTURNON will reconfigure the terminal back to block mode. An example of such use is an application which performs graphics input/output after the terminal has been opened by VOPENTERM and a form has been displayed. VTURNOFF can then be called to prepare for graphics operations. A subsequent VTURNON call will switch the application back to VPLUS/V block mode. The use of VOPENTERM, VCLOSETERM, VTURNON and VTURNOFF provides for a systematic approach to controlling the terminal and driver settings.

The recommended practice for programming VPLUS/V applications is to use only VPLUS/V procedures to perform terminal input/output operations when in VPLUS/V. If the application must perform terminal input/output operations using other input/output methods, either VCLOSETERM or VTURNOFF must be used to switch out of VPLUS/V before doing so.

This simple programming guideline assures that the application will be compatible with the different HP3000 drivers and terminals, and improves its maintainability.

VTURNOFF

Turns off VPLUS/V block mode and enables character mode access without disturbing the terminal screen.

Syntax

VTURNOFF {*comarea*}

Parameters

comarea

Must be *comarea* named when the file was opened by VOPENTERM. If not already set, the following *comarea* items must be set before calling VTURNOFF.

cstatus Set to zero.

comarealen Set to total number of 2-byte words of *comarea*.

VTURNOFF may set the following *comarea* items:

cstatus Set to nonzero value if call is unsuccessful.

fileerrnum Set to file error code if there is an MPE file error.

Discussion

VTURNOFF is used for momentarily switching from VPLUS/V block mode to character mode. This procedure is designed for use after a terminal has been previously opened by VOPENTERM or after a VTURNON.

VTURNOFF reconfigures the terminal and driver for character mode operations without disturbing the screen image on the terminal. The following operations normally performed in VCLOSETERM are not performed in VTURNOFF:

- Clear local form storage
- Enable the USER/SYSTEM keys
- Disable touch reporting, delete touch fields
- Clear screen
- Unlock keyboard
- Close terminal file

VTURNOFF

NOTE

VTURNOFF does not close the terminal file. To close the file and completely reset the driver and the terminal, VCLOSETERM must be used.

Example

The following examples illustrate a call to VTURNOFF:

COBOL

CALL "VTURNOFF" USING COMAREA.

BASIC

200 CALL VTURNOFF(C(*))

FORTRAN

CALL VTURNOFF(COMAREA)

SPL

VTURNOFF(COMAREA);

PASCAL

VTURNOFF(COMAREA);

Turns on VPLUS/V block mode without disturbing the terminal screen.

Syntax

VTURNON {*comarea*,TERMFILE}

Parameters

comarea

The *comarea* name must be unique for each open forms file. The *comarea* must be the same one used in VOPENTERM. The following *comarea* items must be set before the call if not already set:

<i>cstatus</i>	Set to zero.
<i>language</i>	Set to code that identifies the programming language of the calling program.
<i>comarealen</i>	Set to total number of 2-byte words in <i>comarea</i> .

VTURNON may set the following *comarea* fields:

<i>cstatus</i>	Set to nonzero value if call is unsuccessful.
<i>fileerrnum</i>	Set to file error code if there is an MPE file error.
<i>filen</i>	Set to MPE file number of terminal.
<i>identifier</i>	Set to appropriate VPLUS/V terminal ID.
<i>lab'info</i>	Set to appropriate number and length of labels.

termfile

Must be the same terminal file name used in VOPENTERM.

Discussion

VTURNON is normally used in an application when the terminal has already been opened by VOPENTERM, and VTURNOFF was called to switch out of VPLUS/V block mode. VTURNON switches the application back to VPLUS/V block mode without disturbing the image on the terminal screen.

VTURNON

VTURNON reconfigures the terminal and the driver without performing the following operations, which are usually carried out by VOPENTERM:

- Initialize local form storage
- Clear screen
- Enable the USERKEYS function keys
- Disable or enable the USER/SYSTEM key as specified in the SHOWCONTROL word

Unlike VOPENTERM, VTURNON will not ask you to press the BLOCK MODE key if you are using an HP2640B or HP2644 terminal and the terminal is not in block mode.

Example

The following examples illustrate a call to VTURNON:

COBOL
CALL "VTURNON" USING COMAREA, T1.

BASIC
90 T1\$=" "
100 CALL VTURNON(C(*),T1\$)

FORTRAN
T1=" "
VTURNON(COMAREA,T1);

SPL
MOVE T1:=" ";
VTURNON(comarea,T1);

PASCAL
T1:= ' ' ;
VTURNON(COMAREA,T1);

VERSION CONTROL

APPENDIX

H

This appendix provides information on:

- KSAM file control and conversion to MPE files
- Adjust menu
- The HP32209 intrinsic
- The HP32209B utility

KSAM FILE MANAGEMENT

Since the release of version B.03.23 of VPLUS/V, VPLUS/V accepts only MPE files with the file code of VFORM. VPLUS/V has created only MPE files since VPLUS/3000 version A.01.01. Prior to A.01.01, VPLUS/V created KSAM files. Any KSAM file created by VPLUS/V before version A.01.01 will continue to function when accessed through FORMSPEC, REFSPEC, ENTRY or your applications without any actions on your part. If however, you wish to modify the forms file using FORMSPEC or REFSPEC, the file must be converted using the CONVERT utility, as described below.

Purge Forms File

The MPE :PURGE command will delete a non-KSAM forms file. If, however, your forms file is a KSAM file, MPE does not provide a direct means to purge it. If you want to permanently delete a KSAM forms file created through a prior version of FORMSPEC, you must run the KSAMUTIL utility program as follows:

```
:RUN KSAMUTIL.PUB.SYS
```

This program will issue a greater than sign (>) as a prompt. In response to this prompt, you enter the PURGE command. Suppose the forms file you want to purge is named ORDFORM, enter the command as follows:

```
>PURGE ORDFORM  
ORDFORM.PUB.ACCTG & ORDKEY PURGED ← system response
```

Note that you need not specify the key file name associated with your forms file; KSAMUTIL knows this name as it indicates in the response to the PURGE command. KSAMUTIL also notes the account and group in which the forms file resides.

Rename Forms File

If you want to rename an existing non-KSAM forms file, simply use the MPE :RENAME command. In order to rename a KSAM forms file, you must run program KSAMUTIL. You may rename either the forms file or the key file with a single RENAME command; or you may rename both files with two RENAME commands. Suppose you want to rename the forms file ORDFORM and also its associated key file ORDKEY. To do this, run KSAMUTIL and then enter two RENAME commands, as shown:

```
:RUN KSAMUTIL.PUB.SYS  
  
>RENAME ORDFORM, NEWORD  
                                  (new names)  
>RENAME ORDKEY, NEWKEY
```

Returns the version identifier for the installed version of the VPLUS/V intrinsics.

Syntax

HP32209 {*buffer*}

Parameters

buffer Fourteen-byte character string defined in an application to which the version identifier is returned.

Discussion

The version identifier is in the format HP32209*v.uu.ff* where *v* denotes the major enhancement level, *uu* the update level, and *ff* the fix level.

Example

COBOL
CALL HP32209 USING BUFFER

BASIC
200 CALL HP32209(B\$)

FORTRAN
CALL HP32209(BUFFER)

SPL/PASCAL
HP32209(BUFFER)

THE HP32209B UTILITY

The HP32209B utility provides you with a simple way to confirm the version of the VPLUS/V software that you are currently running on the system. HP32209B resides in the PUB.SYS group and account. You can use it by typing the MPE command

```
:RUN HP32209B.PUB.SYS
```

The screen will display the product name and version as well as the version number of each segment of the software. This is to indicate any patches that may have a version number different from the product.

Refer to Appendix J for more information on the use of the MPE/XL version of this utility, called HP32209S, with HP Precision Architecture.

Example

When you run HP32209B, the screen displays the information in this format:

```
VPLUS INTRINSICS VERSION:  HP32209B.05.00
```

SEGMENT	VERSION
01	B'04'22'S01'00
02	B'04'22'S02'00
03	B'04'22'S03'00
04	B'04'22'S04'00
05	B'04'22'S05'00
06	B'04'22'S06'00
07	B'04'22'S07'00
08	B'04'22'S08'00
09	B'04'22'S09'00
10	B'04'22'S10'00
11	B'04'22'S11'00
12	B'04'22'S12'00
13	B'04'22'S13'00

This appendix offers some guidelines for the migration of programs developed on MPE/V to MPE/XL systems, and a short list of peripherals and languages supported on MPE/XL systems.

It also provides information on those aspects of VPLUS/V that pertain exclusively to MPE/XL running on the HP Precision Architecture (HPPA) systems. They are:

- The intrinsics VGETIEEEREAL/VGETIEEEELONG and VPUTIEEEREAL/VPUTIEEEELONG
- The HP32209S utility

MIGRATION ISSUES

Programs that were originally developed on MPE/V must be rewritten to run in native mode on the MPE/XL. For details, refer to the following reference manuals:

- *Migration Process Guide* (Part No. 30367-90007); describes how to migrate existing applications to MPE/XL.
- *Programmers' Skills Migration Guide* (Part No. 30367-90005); explains new program development on MPE/XL and serves as a reference to the operating system.
- *COBOL II/XL Migration Guide* (Part No. 31500-90004); explains how to migrate COBOL II/V programs to MPE/XL.
- *FORTRAN 77/XL Migration Guide* (Part No. 31501-90004); details the migration of FORTRAN 77/V applications to MPE/XL.
- *HP Pascal/XL Migration Guide* (Part No. 31502-90004); details the migration of Pascal applications to MPE/XL. migration of SPL programs to MPE/XL.

Some points to be noted when applications developed on VPLUS/V are run on MPE/XL are described below.

Terminals. When you run an application on MPE/XL, the terminal must be configured for the XON/XOFF handshake. If it is not, large or complex forms will appear to be corrupted when displayed on the screen.

Floating Point Numbers. The type conversion of floating point numbers in VPLUS/V assumes the HP 3000 floating point format, whereas native mode applications store floating point values in IEEE format. The type conversion will work, but the values will be wrong.

Native Mode Pascal Applications. The MPE/XL native mode Pascal compiler allocates 4 bytes for the integer subrange -32768..32767, compared with the 2 bytes allocated by the MPE/V Pascal compiler. You should use type SHORTINT in native mode Pascal applications to get a 2-byte allocation for this subrange.

Record padding in native mode Pascal will cause data misalignment problems for programmers using the VPLUS/V intrinsics, VGETFIELDINFO, VGETFORMINFO, and VGETFILEINFO. The problem can be avoided by packing the *infobuf* records passed as parameters.

Incorrect Parameters. Existing VPLUS/V applications written in COBOL and recompiled into MPE/XL native mode may fail to function. The application either fails with a VPLUS/V error code of 998 or 999, or it aborts with a Memory Fault error. The reason for this is that the programmer has coded the application to pass a display numeric data type as the parameter for a VPLUS/V intrinsic, instead of a binary numeric data type. That is, the COBOL data type PIC 9 has been coded instead of PIC 9 COMP.

This kind of mistake is not caught when the application is compiled, because most COBOL VPLUS/V applications do not use the compiler intrinsic mechanism, "CALL INTRINSIC... USING...". So, the error must be caught at run-time, if at all. It is possible for the defective program to appear to work correctly

in MPE/V and MPE/XL compatibility mode, because VPLUS/V is more forgiving when parameters of the wrong type are passed to it. However, native mode applications on MPE/XL call the VPLUS intrinsics through switch stubs, which require the correct parameter types. The wrong parameter causes a run-time error because the VPLUS native mode access layer does not have the capacity to edit the data passed through it.

If your application fails with a VPLUS error 998 or 999, or a memory fault, you can take the following steps:

1. Activate the VPLUS trace facility and examine the trace, especially the entering and leaving messages from the VPLUS native mode stub software.
2. Check the parameter content messages of the last VPLUS intrinsic call before the program failed.
3. Examine and correct VPLUS parameter declarations as needed, both in native mode code and in compatibility mode code if this is a migrated application.

Supported Peripherals and Applications

The following are currently supported on MPE/XL.

Printers-HPIB	2565, 2566, 2680, 2688
Terminals	2392, 2393, 2394, 2397, 2622, 2624B, 2627, HP 150, Vectra, Port+
Native Mode Languages	COBOL II/XL, FORTRAN 77/XL, HP Pascal/XL
Compatibility Mode Languages	SPL/V, RPG/V, Business BASIC/V, Transact/V

HPPA INTRINSICS

Introduction

On the MPE/V and previous systems, Hewlett Packard represented real or floating point numbers in a format unique to each machine. However, on the MPE/XL systems, these numbers are represented in a format that conforms with an IEEE standard. This means that programs running in native mode on MPE/XL systems will deal with numbers stored in the IEEE format. VPLUS/V will be running in compatibility mode, however, and will thus deal with numbers stored in the MPE/V format.

Currently, VPLUS/V offers intrinsics that will accept a real number in the MPE/V format, convert it to its equivalent external representation and put it into a form's data buffer. The four HPPA intrinsics described below have been developed to allow programs running in native mode to perform the same operations on a number stored in the IEEE format.

NOTE

Use these intrinsics only if you are programming entirely in native mode.
If you are accessing a database in compatibility mode, you must use the
VPLUS/V intrinsics, VGET/PUTtype.

VGETIEEEREAL

Copies character coded data contents from data buffer into an application, converting numeric value to IEEE floating point format.

Syntax



```
VGETIEEEREAL {comarea,fieldnum,variable}
```

Parameters

comarea

Must be the *comarea* name specified when the forms file was opened with VOPENFORMF. If not already set, the following *comarea* items must be set before calling VGETIEEEREAL:

cstatus Set to zero.

comarealen Set to total number of two-byte words in *comarea*.

VGETIEEEREAL may set the following *comarea* items:

cstatus Set to nonzero value if call is unsuccessful or if requested field has an error.

fieldnum

Two-byte integer variable containing the number assigned to the field by FORMSPEC. The field itself may not be longer than 80 characters.

variable

Variable within application, of type IEEE floating point real (32 bits), into which converted value is placed.

For example:

```
VGETIEEEREAL (COMAREA, FIELDNUM, VARIABLE);
```

Discussion

The content of the field denoted by the field number is read from the data buffer. This content must be numeric but its data type need not be; numbers in a field of CHAR type can be converted.

NOTE

The field number is a unique number assigned to each field by FORMSPEC and is completely independent of the field's position in the data buffer or the form. It only changes if the batch command, RENUMBER, is used. You can alter the length, position or other characteristics of a field without

VGETIEEEREAL

affecting the field number. The field number provides a way to locate fields regardless of their position.

The numeric value, stored in the buffer in character coded form, is converted to IEEE floating point format and then copied to the variable in the application. If errors occur during conversion, *cstatus* is set to the appropriate error code. If the requested field's error code is set, its value is moved to the variable but *cstatus* is set to an error code.

VGETIEEEELONG

Copies character coded data contents from data buffer into an application, converting numeric value to IEEE floating point long format (64-bit).

Syntax

VGETIEEEELONG {*comarea*,*fieldnum*,*variable*}

Parameters

comarea Must be the *comarea* name specified when the forms file was opened with VOPENFORMF. If not already set, the following *comarea* items must be set before calling VGETIEEEELONG:

cstatus Set to zero.

comarealen Set to total number of two-byte words in *comarea*.

VGETIEEEELONG may set the following *comarea* items:

cstatus Set to nonzero value if call is unsuccessful or if requested field has an error.

fieldnum Two-byte integer variable containing the number assigned to the field by FORMSPEC. The field itself may not be longer than 80 characters.

variable Variable within application, of type IEEE floating point long (64 bits), into which converted value is placed.

For example:

VGETIEEEELONG(COMAREA, FIELDNUM, VARIABLE);

Discussion

The content of the field denoted by the field number is read from the data buffer. The content of the field must be numeric, but its data type need not be; numbers in a field of CHAR type can be converted.

NOTE

The field number is a unique number assigned to each field by FORMSPEC and is completely independent of the field's position in the data buffer or the form. It only changes if the batch command, RENUMBER, is used. You can alter the length, position or other characteristics of a field without

VGETIEEEELONG

affecting the field number. The field number provides a way to locate fields regardless of their position.

The numeric value, stored in the buffer in character coded form, is converted to IEEE floating point long format and then copied to the variable in the application. If errors occur during conversion, *cstatus* is set to the appropriate error code. If the requested field's error code is set, its value is moved to the variable but *cstatus* is set to an error code.

VPUTIEEEREAL

Writes a floating point number in IEEE standard format from an application to a specified field in the form data buffer, converting the value to character set coded external representation.

Syntax

VPUTIEEEREAL {*comarea*,*fieldnum*,*variable*}

Parameters

comarea Must be the *comarea* name specified when the forms file was opened with VOPENFORMF. If not already set, the following *comarea* items must be set before calling VPUTIEEEREAL:

cstatus Set to zero.

comarealen Set to total number of two-byte words in *comarea*.

VPUTIEEEREAL may set the following *comarea* items:

cstatus Set to nonzero value if call is unsuccessful.

numerrs May be decremented if a new value replaces the old value of a field with an error.

fieldnum Two-byte integer variable containing the number assigned by FORMSPEC to the field in which the value is written. The field itself may not be longer than 80 characters.

variable Variable within application, of type IEEE floating point real (32 bits), that contains the value to be converted to character set coded external representation and copied to a field in the data buffer.

For example: VPUTIEEEREAL (COMAREA, FIELDNUM, VARIABLE);

Discussion

This intrinsic converts an IEEE floating point real number to its character coded form and writes the converted value to a particular field in the data buffer, right justified. The exact format of the written data depends on the type of the destination field. For example, if the number "34.56" were put to a field of type DIG, the result would be "34" since such a field may contain only the digits 0 through 9. The destination field is identified by the field number assigned by FORMSPEC, and must be defined as a numeric field, that is, type NUM, IMP or DIG.

If the specified field had an error, VPUTIEEEREAL will clear the field's error flag and decrement *numerrs*.

NOTE

The field number is a unique number assigned to each field by FORMSPEC and is completely independent of the field's position in the data buffer or the form. It only changes if the batch command, **RENUMBER**, is used. You can alter the length, position or other characteristics of a field without affecting the field number. The field number provides a way to locate fields regardless of their position.

VPUTIEEEELONG

Writes a floating point number in IEEE standard long format from an application to a specified field in the form data buffer, converting the value to character set coded external representation.

Syntax

VPUTIEEEELONG {*comarea*,*fieldnum*,*variable*}

Parameters

comarea

Must be the *comarea* name specified when the forms file was opened with VOPENFORMF. If not already set, the following *comarea* items must be set before calling VPUTIEEEELONG:

cstatus Set to zero.

comarealen Set to total number of two-byte words in *comarea*.

VPUTIEEE long may set the following *comarea* items:

cstatus Set to nonzero value if call is unsuccessful.

numerrs May be decremented if a new value replaces the old value of a field with an error.

fieldnum

Two-byte integer variable containing the number assigned by FORMSPEC to the field to which the value is written. The field itself may not be longer than 80 characters.

variable

Variable within application, of type IEEE floating point long (64 bits), that contains the value to be converted to character set coded external representation and copied to a field in the data buffer.

For example: VPUTIEEEELONG(COMAREA,FIELDNUM,VARIABLE);

Discussion

This intrinsic converts an IEEE floating point long number to its character coded form and writes the converted value to a particular field in the data buffer, right justified. The exact format of the written data depends on the type of the destination field. For example, if the number "34.56" were put to a field of type DIG, the result would be "34" since such a field may contain only the digits 0 through 9. The destination field is identified by the field number assigned by FORMSPEC, and must be defined as a numeric field, that is, type NUM, IMP or DIG.

If the specified field had an error, VPUTIEEEELONG will clear the field's error flag and decrement *numerrs*.

NOTE

The field number is a unique number assigned to each field by FORMSPEC and is completely independent of the field's position in the data buffer or the form. It only changes if the batch command, **RENUMBER**, is used. You can alter the length, position or other characteristics of a field without affecting the field number. The field number provides a way to locate fields regardless of their position.

THE HP32209S UTILITY

The HP32209S utility provides you with a simple way to confirm the versions of both the native mode software and the compatibility mode software that you are currently running on the system. HP32209S resides in the PUB.SYS group and account. You can use it by typing the MPE command

```
:RUN HP32209S.PUB.SYS
```

The screen will display the product name and version as well as the version number of each segment of the software. This is to indicate any patches that may have a version number different from the product.

This is a quick way to test whether native mode is moving through to compatibility mode on the MPE/XL system, and hence to confirm that the VPLUS/V software is accessible in compatibility mode.

Example

When you run HP32209S, the screen displays the information in this format, first for native mode and then for compatibility mode:

```
VPLUS INTRINSICS VERSION:  HP32209S.05.00
```

```
-----
SEGMENT    VERSION
-----
01      B'04'22'S01'00
02      B'04'22'S02'00
03      B'04'22'S03'00
04      B'04'22'S04'00
05      B'04'22'S05'00
06      B'04'22'S06'00
07      B'04'22'S07'00
08      B'04'22'S08'00
09      B'04'22'S09'00
10      B'04'22'S10'00
11      B'04'22'S11'00
12      B'04'22'S12'00
13      B'04'22'S13'00
-----
```


A

- Access, using VPLUS/V intrinsics, 6-28
- Access, method errors, B-2
- Add Check Digit option, REFSPEC, 5-43
- ADDARBFIELD, description, 7-19b
- Adding,
 - a form, FORMSPEC, 3-3, 3-8, 3-37
 - a reformat, REFSPEC, 5-19
 - a save field, FORMSPEC, 3-38
- Adjust Menu, H-4
- Advanced,
 - forms design, overview, 4-1
 - levels of, 4-2
 - processing, overview, 1-6, 1-11, 4-1
 - specifications, 4-36
 - statements, 1-13
- Alphabetic check digits, calculation, D-4
- Alter forms sequence, FORMSPEC, 4-50
- Appended forms,
 - ENTRY, 2-9
 - FORMSPEC, 3-18
 - with local form storage, 6-26
- Application-ready Buffer, 1-13, 3-53a
 - batch mode commands, 7-4, 7-19a
 - benefits, 1-13a
 - creating, 3-3, 3-7, 3-39
 - data types, 3-15
 - generating, 3-53d
 - trace facility, 3-53i
 - use in VGETBUFFER, 6-44
 - using and defining, 3-53j
- Applications hints, E-1
- Applications programmer, using manual, 1-4
- Applications user, 1-5
- ARB, see Application-ready Buffer
- ARBTOSCREEN, description, 7-19c
- Arithmetic calculation, 1-13
- Arithmetic expressions, FORMSPEC syntax, 4-11
- ASCII, see USASCII
- Assigning value to field, FORMSPEC, 4-38
- Automatic data formatting, FORMSPEC, 4-41
- AUTOREAD, 6-19, 6-109, G-11

B

- Bar Code Reader, 3-32
- BASIC,
 - interface to VPLUS/V, A-1

- intrinsic call, 6-7
- parameters, 6-8
- sample program, A-58
- Batch file,
 - close, VPLUS/V intrinsic, 6-34
 - concatenating, 5-1, 5-45
 - creating, VPLUS/V intrinsic, 6-75
 - current record number, see RECNUM
 - deleting a record, VPLUS/V intrinsic, 6-115
 - expanding, 2-7
 - number of records, see RECNUM
 - open, VPLUS/V intrinsic, 6-81
 - protection against system crash, 5-3, 6-92
 - read, VPLUS/V intrinsic, 6-106
 - relation to other files, 5-2
 - REFSPEC, 5-1
 - Removing deleted records, 2-7
 - specification in ENTRY, 2-5
 - write to, VPLUS/V intrinsic, 6-124
- Bar code,
 - Defining from Data Capture Device Specifications Menu, 3-57
 - Defining from CONFIG phase, 4-7
- Batch mode, FORMSPEC in, 7-1
 - commands, 7-3
 - errors, B-12
- Batch record, read, 6-94
- Bit Map conversion, I-1
- Blank replacement,
 - in output, REFSPEC, 5-25
 - FORMSPEC, 4-48
- Blink,
 - FORMSPEC, 3-14, 3-46
 - VPLUS/V intrinsic code, 6-13
- BLOCK MODE,
 - use in ENTRY, 2-8
 - use in FORMSPEC, 3-2
- Bracketed field, ENTRY, 2-11
- Brackets, field delimiters, FORMSPEC, 3-9
- BREAK key,
 - recovery from in,
 - ENTRY, 2-24
 - FORMSPEC, 3-7
 - REFSPEC, 5-13
 - steps to recovery, G-14
- Browse data, VPLUS/V intrinsic, 6-94
- BROWSE/COLLECT key, ENTRY, 2-3
- BROWSE/MODIFY
 - key, ENTRY, 2-16
 - requesting mode, 2-16
 - use of, 1-18
- Buffer,
 - configuration, terminal, G-13
 - control for ARB, 6-19, 6-43
 - data area, 6-4, 6-6

length, see DBUFLEN
 read data from, VPLUS/V intrinsic, 6-43
 write data to, VPLUS/V intrinsic, 6-86
 BUILD command, with forms files, 3-27
 Byte array, VPLUS/V intrinsic, 6-8

C



Calling VPLUS/V procedures, 6-7
 Carriage return, block mode, 2-8
 CDIGIT, Edit Statement, FORMSPEC, 4-29
 Center data, FORMSPEC, 4-52
 CFNAME, definition, 6-9, 6-14
 CFNUMLINES, definition, 6-15
 Change,
 a field or form, FORMSPEC, 3-8, 3-25
 forms file name, REFSPEC, 5-20
 statement, FORMSPEC, 4-9, 4-50
 CHAR type, FORMSPEC, 3-15
 Character,
 comparison, FORMSPEC, 4-23
 constants, specification of, 4-10
 data types, FORMSPEC, 3-15
 deletion,
 FORMSPEC, 4-4
 REFSPEC, 5-38
 formatting, FORMSPEC, 4-49
 set, USASCII, C-1
 string, data type, 6-8
 Check digit,
 adding, REFSPEC, 5-43
 alphabetic, D-6
 calculation, D-1
 validation, 4-29
 validation, FORMSPEC, 4-29
 Choice pattern, MATCH statement, 4-31
 Close, VPLUS/V procedure for,
 batch file, 6-34
 forms file, 6-35
 terminal, 6-36
 CMODE, definition, 6-12
 COBOL,
 interface to VPLUS/V, A-1
 parameters, 6-8
 sample program, A-26
 Coding, Touch Feature, E-7
 Collating sequence, USASCII characters, C-1
 Collection of data, simple, 1-6
 COMAREA,
 definition, 6-9
 extension, 6-12
 initialization, 6-12
 size of extension, 6-12

- terminal information, G-11
- words, 6-12
- COMAREALEN, definition, 6-12
- Combining data in forms, REFSPEC, 5-7
- Command file, 7-2, 7-20
- Commas, in numeric data, FORMSPEC, 3-15
- Comments,
 - FORMSPEC field processing, 4-5
 - in FORMSPEC forms, 3-42
 - use of in command files, 7-3
 - use of in forms families, 3-31
- Communication area, 6-9, 6-11
- Comparisons, single value, 4-23
- Compiling,
 - forms file, 3-27
 - FORMSPEC in batch mode, 7-3
- Concatenating Batch Files, 5-45
- Conditional processing, FORMSPEC, 4-52
- Condition, FORMSPEC IF statement, 1-13, 4-52
- CONFIGuration statements, 4-7, 4-13
- Constant,
 - as field processing operand, 4-10
 - output file, REFSPEC, 5-32
- Continuing lines, field processing, 4-5
- Control statements, FORMSPEC, 4-9
- Conversion,
 - of data, REFSPEC, 5-35
 - in data movement, FORMSPEC, 4-43
 - numeric, VPLUS/V intrinsic, 6-65
- Convert,
 - buffer data, 6-65
 - output data to EBCDIC, REFSPEC, 5-25
- Converting KSAM to MPE forms files, H-3
- Copying a form,
 - FORMSPEC, 3-25, 3-39
 - FORMSPEC in batch mode, 7-1
- Correcting errors, ENTRY, 2-14
- Create batch file, 6-75
- CSTATUS, definition, 6-12
- Custom Error Message, 4-5

D

- Data buffer,
 - area, 6-4
 - length, see DBUFLEN
 - read, VPLUS/V intrinsic, 6-43
 - write to, VPLUS/V intrinsic, 6-86
- Data capture devices,
 - COMAREA items, 6-11
 - configuration phase, 4-13, 4-59
 - Data Capture Device Specifications Menu, 3-56
 - special features, 1-8, G-1

- Terminal/Language Selection Menu, 3-54, 6-5, 6-115
- Data Capture Device Specifications Menu, data capture devices, 3-56
- Data collection,
 - resumption in ENTRY, 2-16
 - use of, 1-17
- Data conversion,
 - FORMSPEC, 4-43
 - REFSPEC, 5-38
- Data editing,
 - simple, 1-8
 - VPLUS/V intrinsic, 6-44
- Data entry,
 - ENTRY, 2-1, 2-12
 - program, 1-17
 - VPLUS/V procedures, 6-40
- Data formatting,
 - FORMSPEC, 4-41
 - REFSPEC, 1-13, 5-36
- Data interpretation, FORMSPEC, 3-16
- Data justification, FORMSPEC, 5-56
- Data modification,
 - ENTRY, 2-16, 2-17
- Data movement, FORMSPEC, 1-13, 4-40
- Data processing, VPLUS/V procedures, 6-31
- Data reformatting,
 - REFORMAT, 5-1
 - overview, 1-19
- Data type,
 - ARB, 3-17a
 - conversion, FORMSPEC, 4-43
 - formatting, REFSPEC, 5-36
 - FORMSPEC field, 3-15
 - REFSPEC output field, 5-35
 - save fields, 3-53
 - screen, 3-15
 - use of, 1-9
 - VPLUS/V procedures, 6-8
- Data,
 - deletion, ENTRY, 2-7, 2-17
 - modify, VPLUS/V intrinsic, 6-115
 - read from buffer, VPLUS/V intrinsic, 6-43
 - source, 1-1
 - viewing it in ENTRY, 2-17
 - write to batch file, VPLUS/V intrinsic, 6-115
 - write to buffer, VPLUS/V intrinsic, 6-86
- Date,
 - comparisons, FORMSPEC, 4-24
 - constant, specification of, 4-10
 - conversion, FORMSPEC, 4-37
 - data types, FORMSPEC, 3-16
 - formatting, FORMSPEC, 4-42
 - type conversion, 3-53c
- Data Conversions, define, 3-50
- Date fields, interpretation, FORMSPEC, 3-17

- FORMSPEC, 3-4
- REFSPEC, 5-14
- Expanding files,
 - batch, 2-7
 - forms, 3-27
- Expressions, arithmetic, FORMSPEC syntax, 4-11
- Extension of COMAREA, BASIC, 6-12

F

- FAIL statement, FORMSPEC, 4-55
- Families of forms, 3-30
- Fast forms file, FORMSPEC, 3-39
- FCOPY,
 - use with ENTRY FILES, 2-7
 - with forms files, 3-27
- Features, VPLUS/V, 1-2
- Field,
 - delimiters,
 - ENTRY, 2-11
 - FORMSPEC, 3-9
 - edit,
 - overview, 4-20
 - phase, FORMSPEC, 4-60
 - statements, summary, 4-20
 - use of, 1-6, 1-10
 - VPLUS/V intrinsic, 6-39
 - enhancement,
 - FORMSPEC, 3-14
 - use of, 1-9
 - formatting output, 5-35
 - heading, FORMSPEC, 4-57
 - information, obtaining, 6-48
 - length,
 - as constant value, 4-10, 4-21
 - FORMSPEC, 3-12
 - REFSPEC output field, 5-32
 - save fields, 4-14
 - menu,
 - custom error message example, 4-5
 - FORMSPEC, 3-22
 - FORMSPEC processing specifications, 4-3
 - REFSPEC output, 5-34
 - sample of, 1-9
 - name,
 - as field processing operand, 4-9
 - FORMSPEC, 3-13
 - REFSPEC input field, 5-27, 5-31
 - REFSPEC output field, 5-31
 - Required/Optional, 5-31
 - save fields, 3-53
 - use of, 1-9
 - number,

FORMSPEC, 3-12
 VPLUS/V procedures, 6-47, 6-88
 processing specifications, 4-20
 correcting, 4-4
 entering, 4-3
 overview, 4-2
 processing statements,
 conditional, FORMSPEC, 4-52
 summary, 4-7
 REFSPEC output, 5-34
 separator, REFSPEC, 5-26
 specifications, 5-6
 reformat file, 5-3
 REFSPEC, 5-3
 standard formatting, 5-36
 substring 5-31
 symbol, table errors, B-6
 tag, FORMSPEC, 3-11
 TOGGLE key, FORMSPEC, 3-4
 type, use of, 1-8
 write to buffer, VPLUS/V intrinsic, 6-88
 Fields,
 entering data, 2-11
 maximum number on one form, 3-9
 protected, definition of, 2-11
 read from terminal, VPLUS/V intrinsic, 6-45
 unprotected, definition of, 2-11
 File,
 batch mode command, 7-10
 command,
 with ENTRY file, 2-7
 with forms files, 3-27
 error number (FCHECK code), see **FILERRNUM**
 errors, 6-22
 number,
 error message file, see **ERRFILENUM**
 print file, see **ERRFILENUM**
 File Equations,
 FORMSPEC, 7-15, E-3
 ENTRY, Intrinsics, E-4
 REFSPEC, 5-44
FILEN, definition of, 6-18
FILERRNUM, definition, 6-17
 Files,
 ENTRY, 2-5
 for REFORMAT program, 5-3
 FORMSPEC,
 compiled forms file, 3-27
 fast forms file, 3-39
 forms file, 3-25
 REFSPEC, 5-3
FILL,
 option, REFSPEC, 5-42
 statement, FORMSPEC, 4-48

- Finish,
 - heading, FORMSPEC, 4-57
 - phase processing, VPLUS/V intrinsic, 6-41
- First record marker, REFSPEC output file, 5-32
- Float sign, REFSPEC, 5-40
- Floating decimal point, FORMSPEC, 3-15
- Forced,
 - data formatting, FORMSPEC, 4-44
 - failure, FORMSPEC edits, 4-55
- Form,
 - adding a, 2-9, 3-8, 3-37
 - appended, FORMSPEC, 3-18, 6-26
 - copy a, 3-25, 3-39, 7-1
 - definition area, 6-4
 - deleting a, 3-25, 3-38, 7-1
 - design,
 - advanced, overview, 4-1
 - processing, 1-11
 - FORMSPEC, 3-8
 - levels, of, 1-6
 - preparation for, 3-8
 - resuming, 3-7
 - display at terminal, VPLUS/V intrinsic, 6-108
 - families, 4-71, 6-26
 - function key labels, 3-43
 - frozen,
 - ENTRY, 2-9
 - FORMSPEC, 3-18, 3-41
 - initialize, VPLUS/V intrinsic, 6-68
 - layout,
 - example of, 1-8
 - FORMSPEC, 3-3, 3-44
 - modification, 3-8
 - planning for, 3-22
 - listing of, 3-28, 3-38
 - loading, 6-25
 - local storage, 2-8, 6-24, 6-26
 - menu, example, 1-7
 - modification, FORMSPEC, 3-8
 - name,
 - current, FORMSPEC, 3-41
 - current, FORMSPEC in batch mode, 7-10
 - head form, 3-41, 3-49
 - next, FORMSPEC, 3-41
 - print with, VPLUS/V intrinsic, 6-84
 - printing of in ENTRY, 2-4
 - processing,
 - advanced, 1-12
 - conditional, 4-52
 - phases, 4-57
 - STOP statement, 4-56
 - purging from local storage, 6-27
 - repeating,
 - ENTRY, 2-9

- FORMSPEC, 3-18, 3-41
- sequence, 1-13
 - altering, 4-50
- ENTRY, 2-9
- REFSPEC input file, 5-27
- storage directory, 6-25
- Format, VPLUS/V intrinsic calls, 6-7
- Formatting,
 - by data type, REFSPEC, 5-36
 - by command,
 - FILL, 4-48
 - JUSTIFY, 4-47
 - SET, 4-41
 - STRIP, 4-46
 - UPSHIFT, 4-49
 - character, FORMSPEC, 4-41
 - date, FORMSPEC, 4-42
 - numeric, FORMSPEC, 4-41
 - output data, REFSPEC, 5-35
 - specifications,
 - FORMSPEC, 4-50
 - REFSPEC, 5-38
- Forms and output records, REFSPEC, 5-5
- Forms designer, using manual, 1-4
- Forms file,
 - close, VPLUS/V intrinsic, 6-35
 - compiled version, 3-27
 - compilation, 3-27, 3-39
 - design, E-2
 - fast version, 3-39
 - layout of, 3-25
 - menu,
 - FORMSPEC, 3-2, 3-35
 - REFSPEC, 5-18
 - name,
 - altering, REFSPEC, 5-20
 - changing in FORMSPEC, 3-2, 3-35
 - changing in REFSPEC, 5-20
 - reformat file, 5-44
 - REFSPEC, 5-3
 - open, VPLUS/V intrinsic, 6-76
 - printing, 3-21
 - sequencing, 1-13, 2-9
 - source version, 3-25
 - specification,
 - in ENTRY, 2-5
 - in FORMSPEC, 3-2, 3-35
 - in REFSPEC, 5-18
- Forms files,
 - converting KSAM to MPE, H-3
 - expanding, 3-27
 - in batch mode FORMSPEC, 7-1
 - obtaining information, 6-54
 - printing, 3-28, 3-38



- purging, H-2
 - recompiling, 3-27, 3-39
 - renaming, H-2
- Formsloading error, B-57
- FORMSPEC,
 - adjust menu, H-4
 - advanced forms design, overview, 4-1
 - advanced processing, overview, 4-2
 - apply errors, B-20
 - batch mode
 - errors, B-12
 - overview, 7-1
 - error messages list, B-2
 - field edits, 1-10
 - form menu, 1-7
 - function keys, 3-4
 - hints for using, E-1
 - init and compile, B-16
 - menu controller errors, B-13
 - menus, 1-7, 3-5
 - parser errors, B-20
 - program,
 - executing, 3-2
 - overview, 3-1
 - terminating, 3-7
 - use of, 1-6
 - scanner errors, B-18
 - screen design, 1-8
- FORMSTORESIZE, definition, 6-17
- FORTRAN,
 - interface to VPLUS/V, overview, 1-14, 6-1, A-1
 - parameters, 6-8
 - sample program, A-38
- Freeze form, VPLUS/V intrinsic code, 6-15
- Freeze/Append option, FORMSPEC, 3-18, 3-41
- FREEZAPP, definition, 6-15
- Frozen forms,
 - ENTRY, 2-9
 - FORMSPEC, 3-18, 41
- Function Key Labels,
 - Accessing by intrinsics, 6-58, 6-104
 - Defining labels, 3-43, 3-51
- Function keys,
 - ENTRY, 2-3, 2-4
 - FORMSPEC, 3-4
 - define labels for, 3-43, 3-51
 - read, VPLUS/V intrinsic, 6-96
- REFSPEC, 5-14
- sequence control,
 - in FORMSPEC, 3-18
 - in REFSPEC, 5-14

G

GE statement, FORMSPEC, 4-23
 GENARB, description, 7-19f
 Get,
 next form, 6-62
 with VPLUS/V procedures, 6-62
 data from memory, 6-43
 form from forms file, 6-62
 Global,
 adding specifications, 5-20
 defining in REFSPEC, 5-24
 information, reformat file, 5-3
 specification,
 FORMSPEC, 3-48
 REFSPEC, 5-20
 use of, 1-13
 Globals,
 define function key labels, 3-51
 menu,
 FORMSPEC, 3-48
 REFSPEC, 5-24
 Grouping, MATCH statement, 4-32
 GT statement, FORMSPEC, 4-23

H

Half bright,
 FORMSPEC, 3-14, 3-46
 VPLUS/V intrinsic code, 6-13
 Head form,
 key, ENTRY, 2-3
 name, FORMSPEC, 3-27
 Header messages, B-66
 Hints for using FORMSPEC, E-1
 HP32209B intrinsic, H-5
 HP32209B utility, H-6
 HP32209S utility, J-11

I

IF statement, FORMSPEC, 4-52
 Implied decimal point, FORMSPEC, 3-15, 3-47
 IMPn field, FORMSPEC, 3-15, 3-47
 IN statement,
 FORMSPEC,
 range check, 4-27
 table check, 4-25
 Index retrieve operand, FORMSPEC syntax, 4-12
 Information procedures, 6-48
 INIT heading, FORMSPEC, 4-57
 Initial,

- phase processing, VPLUS/V procedures, 6-68
- value,
 - in field, FORMSPEC, 3-17, 3-47
 - save field, 3-53
 - use of, 1-10
- Initialization phase, FORMSPEC, 4-58
- Initialize,
 - COMAREA, 6-12
 - form, VPLUS/V intrinsic, 6-68
- Input,
 - field specification, REFSPEC, 5-27
 - FORMS,
 - changing menu, 5-20
 - menu, REFSPEC, 5-27
 - sequence modifying, 5-27
 - sequence, reformat file, 5-5
 - sequence, specification of, 5-36
- Integer,
 - convert,
 - from USASCII, VPLUS/V intrinsic, 6-64
 - to USASCII, VPLUS/V intrinsic, 6-90
 - VPLUS/V intrinsic call, 6-8
- Interface to programs, use of, 1-14
- Interpreting entered data, FORMSPEC, 3-16
- Interrupting,
 - data entry, ENTRY, 2-15
 - forms design, 3-7
 - recovery, G-14
 - REFSPEC, 5-13
- Intrinsics
 - calling, 6-7
 - dependency, 6-31
 - description of, 6-29
 - flow 6-29
 - operation, 6-5
 - summary of, 6-2
 - using, 6-1
- Inverse video,
 - FORMSPEC, 3-14, 3-46
 - VPLUS/V intrinsic code, 6-13

J

- JUSTIFY,
 - option, REFSPEC, 5-39
 - statement, FORMSPEC, 4-47

K

- Keyboard, terminal, 2-2
- Key definitions, REFSPEC, 5-14
- Key label area of memory, 6-6

KSAM file management,
 convert, H-3
 purge, H-2
 rename, H-2
 KSAM forms files, 2-6, 3-36, H-2

L

Label, batch file, 6-74
 Label option, COMAREA definition, 6-14
 LANGUAGE, COMAREA definition, 6-11
 LASTKEY, COMAREA definition, 6-13
 LE statement, FORMSPEC, 4-23
 Left justify,
 data, FORMSPEC, 4-47
 output data, REFSPEC, 5-39
 Length check, FORMSPEC edits, 4-21
 Levels of advanced forms design, 4-2
 LIGHT, configuration command, 4-15
 LIGHTSON definition of, 6-20
 Lines, continuation of, in field processing, 4-5
 Lines per page, adjustment, E-5
 Listing,
 forms, FORMSPEC, 3-28, 3-38
 forms, FORMSPEC in batch mode, 7-1
 reformat, REFSPEC, 5-21
 reformatted data, REFORMAT, 5-21
 Local form function key labels, 3-43
 form families, 6-26
 Local form storage,
 form families, 6-26
 form storage directory, 6-25
 FORM'STORE'SIZE, 6-17, 6-76
 HP 2624B terminal, 6-17, 6-24, G-3
 HP 2626A terminal, 6-24, G-3
 HP 2394A terminal, 6-24, G-3
 loading forms, 6-25
 look-ahead loading, 6-26
 LOOK'AHEAD, 6-15, 6-26, 6-96
 overview, 6-24
 preloading, 6-25
 purging, 6-26
 SHOWCONTROL, 6-16, 6-26, 6-108
 VLOADFORMS, 6-26, 6-70
 VUNLOADFORM, 6-112
 with appending forms, 6-27
 with ENTRY, 2-8
 with form families, 6-26
 LOCALEDITS, configuration command, 4-16
 Lockword,
 for forms file, 3-36
 on reformat file, 5-13
 Logic errors, ENTRY, 2-14

Logical, data type, 6-8
 Long data type,
 convert,
 from USASCII, VPLUS/V intrinsic, 6-64
 to USASCII, VPLUS/V intrinsic, 6-90
 usage, 6-8
 Look-ahead loading, 6-26
 LOOK'AHEAD, definition, 6-15
 LT statement, FORMSPEC, 4-23

M

Main Menu,
 FORMSPEC, 3-37
 REFSPEC, 5-19
 MAIN/RESUME key,
 FORMSPEC, 3-4
 REFSPEC, 5-14
 Manual, use of, 1-4
 MATCH,
 operator hierarchy, FORMSPEC, 4-34
 statement, FORMSPEC, 4-30
 MDY field, 3-16
 Memory, use by VPLUS/V procedures, 6-4
 Menu,
 definition, 2-9
 FORMSPEC,
 Data Capture Device Specifications, 3-56
 Default formatting, 4-41
 Field,
 description, 3-11, 3-45
 with processing specifications, 4-3
 with custom error messages, 4-5
 Form, 3-18, 3-40
 Form Function Key Labels, 3-43
 Forms File, 3-35
 Globals, 3-48
 Global Function Key Labels, 3-51
 INIT and Processing errors, B-7
 Main, 3-37
 Processing Table errors, B-7
 sample form designs, 3-23
 Save Field, 3-52
 Terminal/Language Selection, 3-54
 REFSPEC,
 Forms File, 5-18
 Globals, 5-24
 Input Forms, 5-27
 Main, 5-19
 Output Field, 5-34
 Output Record, 5-29, 5-30
 Reformat File, 5-17
 Menus,

- ease of forms design, 3-8
- FORMSPEC program, 3-5
- REFSPEC program, 5-15
- sequence in,
 - FORMSPEC, 3-5
 - REFSPEC, 5-15
- Message,
 - display at terminal, VPLUS/V intrinsic, 6-108
 - error return, VPLUS/V intrinsic, 6-37
 - FORMSPEC custom error, 4-5
 - move to window, VPLUS/V intrinsic, 6-92
- MINLEN statement, FORMSPEC, 4-21
- Mixing printing and nonprinting delimiters, 3-10
- MODARB, description, 7-19g
- Modifying,
 - data, ENTRY, 2-1, 2-16
 - field processing specifications, 4-4
 - form, FORMSPEC, 3-25
 - forms file, FORMSPEC, 3-25
 - reformat, REFSPEC, 5-20
- Modulus 10, check digit calculation, D-2
- Modulus 11, check digit calculation, D-3
- MOVEARBFIELD, description, 7-19h
- Movement of data, FORMSPEC, 4-40
- MPE/XL, migration issues, J-3
 - native mode intrinsics, J-4
 - Pascal applications, J-2
 - supported peripherals, J-3
 - terminals, J-2
- MULTIFUNCTION READER, 3-57
- Multilanguage, 6-1
- Multipurpose, 6-1
- Multiusage, definition, 6-14

N

- Naming files, ENTRY, 2-6
- Native Language Support,
 - date handling, 4-28, 8-3
 - ENTRY, 2-6, 8-4
 - errors, B-67
 - field edits, 8-3
 - international forms file, 8-2
 - intrinsics,
 - VGETLANG, 6-2, 6-60, B-55
 - VSETLANG, 6-2, 6-106, B-55
 - language attribute, 8-1
 - MATCH, 4-30
 - NATIVE-3000, 4-24, 8-1
 - numeric data, 3-16, 3-47, 8-4
 - overview, 8-1
 - pattern matching, 4-30
 - range checks, 4-28

- setting language ID number, 8-2
- single value comparisons, 4-24, 8-3
- table checks, 4-25
- Terminal/Language Selection Menu, 3-54
- upshifting, 4-49
- NE statement, 4-23
- NEXT FORM key,
 - ENTRY, 2-3
 - FORMSPEC, 3-4
- Next form name, FORMSPEC, 3-41
- NEXT key,
 - FORMSPEC, 3-4
 - REFSPEC, 5-14
- NEXT REFORMAT key, REFSPEC, 5-14
- NFNAME, definition, 6-14
- NIN statement,
 - range check, 4-27
 - table check, 4-27
- Nonprinting delimiters, 3-9
- Number of,
 - errors, set by VPLUS/V intrinsic, 6-99
 - lines in current form, see CFNUMLINES
 - records in batch file, see NUMRECS
- Numeric,
 - comparisons, FORMSPEC, 4-24
 - constants, specification of, 4-10
 - conversion,
 - FORMSPEC, 4-43
 - VPLUS/V intrinsic, 6-64, 6-90
 - data types, FORMSPEC, 3-15, 3-47
 - formatting, FORMSPEC, 4-41
- NUMERRS, definition, VPLUS/V intrinsic, 6-13
- NUM, standard formatting, 5-37
- NUMn field, FORMSPEC, 3-15, 3-47
- NUMRECS, definition, VPLUS/V intrinsic, 6-18

O

- Open,
 - batch file, VPLUS/V intrinsic, 6-72
 - forms file, VPLUS/V intrinsic, 6-76
 - terminal, VPLUS/V intrinsic, 6-78
- Operands, field processing statements, 4-9
- Operands, hierarchy, 4-34
- Operators, MATCH statement, 4-34
- Optional,
 - field,
 - ENTRY, 2-12
 - FORMSPEC definition, 3-14
 - specification, MATCH statement, 4-32
- Output Field Menu, REFSPEC, 5-34
- Output field specification,
 - reformat file, 5-6

- REFSPEC output record, 5-34
- Output file,
 - name, REFORMAT, 5-44
 - REFSPEC, 5-3
- Output record,
 - definition, 5-6
 - changing, 5-21
 - modifying, 5-30
 - reformat file, 5-3
 - REFSPEC, 5-6
- Output Record Menu, REFSPEC, 5-29

P

- Parameter types, VPLUS/V intrinsic, 6-8
- Parent form, 3-30
- PASCAL,
 - interface to VPLUS/V, overview, 1-14, A-1
 - parameters, 6-6
 - sample program, A-75
- Pattern match, FORMSPEC edit, 4-30
- Performance, with form families, 3-30
- Phase specification statements, FORMSPEC, 4-57
- Phases of form processing, FORMSPEC, 1-13, 4-57
- Preloading, 6-25, 6-108
- Preparing for forms design, 3-22
- PREVIOUS FORM key, FORMSPEC, 3-4
- PREVIOUS key,
 - ENTRY, 2-3
 - FORMSPEC, 3-4
 - REFSPEC, 5-14
- Print,
 - file number, see PRINTFILENUM
 - form, VPLUS/V intrinsic, 6-84
- PRINT key, ENTRY, 2-3
- Printed delimiters, 3-9
- PRINTFILNUM, definition, 6-17
- Printing data and forms, ENTRY, 2-4
- Process field, FORMSPEC definition, 3-14
- Processing specification, how to enter, 4-3, 4-13
 - Advanced Processing statement, 1-12, 1-13
 - Processing statement, 4-7
- Program generated data, ENTRY, 2-13
- Program,
 - interface, 1-15
 - interruption,
 - ENTRY, 2-15
 - FORMSPEC, 3-7
 - recovery, G-14
 - REFSPEC, 5-13
 - terminating,
 - ENTRY, 2-15
 - FORMSPEC, 3-7

REFSPEC, 5-13

Programmatic interface, use of, 1-14
 Programmer, using manual, 1-4
 Programming in BASIC, COMAREA extension, 6-12
 Programs, samples using VPLUS/V intrinsics, A-1
 Protected fields, definition of, 2-1
 Protecting batch file from system crash, 6-82
 Purging forms file, H-2
 Put, field in buffer, VPLUS/V intrinsic, 6-88

R

Range checks, FORMSPEC, 4-27
 Read and convert field, VPLUS/V intrinsic, 6-64, 6-90
 Read,
 batch record, VPLUS/V intrinsic, 6-94
 data buffer, VPLUS/V intrinsic, 6-43
 data from terminal, VPLUS/V intrinsic, 6-96
 field,
 and convert from USASCII, VPLUS/V intrinsic, 6-64
 from memory, VPLUS/V intrinsic, 6-45
 form from forms file, VPLUS/V intrinsic, 6-62
 Real value,
 convert from USASCII, VPLUS/V intrinsic, 6-64
 convert to USASCII, VPLUS/V intrinsic, 6-90
 VPLUS/V intrinsic call, 6-8
 RECNUM, definition, 6-18
 Recompiling forms file, 3-27
 Record,
 definition, reformat file, 5-3
 delete from batch file, VPLUS/V intrinsic, 6-114
 format,
 batch file, 6-73
 output file, REFSPEC, 5-24
 length, output file, REFSPEC, 5-25
 marker, REFSPEC, 5-6, 5-32
 number,
 batch file, see RECNUM
 terminator, REFSPEC, 5-25
 read from batch file, VPLUS/V intrinsic, 6-94
 REFSPEC output definition, 5-31
 write to batch file, VPLUS/V intrinsic, 6-114
 Records in batch file, see NUMRECS
 Recovery from program interruption,
 ENTRY, 2-15
 FORMSPEC, 3-7
 REFSPEC, 5-13
 steps to, G-14
 REFLIST file, REFORMAT, 5-45
 Reformat,
 adding to reformat file, 5-19
 deletion of, 5-21
 design, resuming, 5-21

- error messages, list of, B-57, B-63
- file,
 - creates MPE, 5-3
 - compiling, 5-19
 - diagram of, 5-4
 - name,
 - REFORMAT, 5-44
 - REFSPEC, 5-19
 - REFSPEC, 5-3
 - specification, REFSPEC, 5-17
- listing, 5-21
- modification of, 5-20
- Reformat File Menu, REFSPEC, 5-17
- REFORMAT program,
 - executing, 5-44
 - streaming, 5-46
 - use of, 5-1
- Reformatting,
 - data, 2-1
 - capability, 1-19
 - conversion and movement, 5-36
 - REFORMAT, 2-1
 - order of, 5-35
 - repeating forms, REFSPEC, 5-10
 - overview, 5-1
 - standard 5-36
- REFRESH key,
 - ENTRY, 2-3
 - FORMSPEC, 3-4
 - REFSPEC, 5-14
- REFSPEC,
 - Access Method, B-24
 - error messages, list of, B-24
 - field table errors, B-30
 - function keys, 5-17
 - interruption, 5-13
 - menu processing, B-32
 - menus, 5-20
 - output sequence definition, B-25
 - program,
 - executing, 5-13
 - overview, 5-1
 - terminating, 5-13
 - use of, 1-18, 5-13
 - standard formatting, 5-36
 - validation error, B-28
- RELATE command,
 - use, 7-16
 - errors, B-14
- Relating forms,
 - FORMSPEC, 3-31, 3-39, 3-42
 - FORMSPEC in batch mode, 7-1, 7-4, 7-16
 - REFSPEC, 5-5
- Remote terminals, via DS facility, E-3

- Removing deleted records, 2-7
- RENAMEARBFIELD, description, 7-19i
- Repeat option, FORMSPEC form definition, 3-18, 3-41
- REPEATAPP, definition, 6-15
- Repeating form,
 - ENTRY, 2-9
 - FORMSPEC, 3-18, 3-41
 - reformatting, 5-10
 - VPLUS/V intrinsic code, 6-15
- Repetition, MATCH statement, 4-33
- Replace blanks,
 - in data, FORMSPEC, 4-48
 - in output, REFSPEC, 5-42
- Reproduced From field, 3-42
- Required field,
 - ENTRY, 2-12
 - FORMSPEC definition, 3-14
- Reserved word list, FORMSPEC, 3-13
- Resuming,
 - data collection, ENTRY, 2-16
 - design, 5-21
 - execution, FORMSPEC, 3-7
 - forms design, FORMSPEC, 3-25
 - reformat design, REFSPEC, 5-21
- Retries, definition of, 6-18
- Right justify,
 - data, FORMSPEC, 4-47
 - output data, REFSPEC, 5-39
- Root form, 3-30
- Running,
 - ENTRY, 2-5
 - REFORMAT, 5-44
 - REFSPEC, 5-13

S

- Sample batch data entry programs, A-1
- Sample command file, 7-2
- Save field,
 - adding to forms file, 3-38
 - data type, specification of, 3-53
 - deleting from forms file, 3-38
 - impossible to copy, 3-25
 - length, specification of, 3-53
 - menu, FORMSPEC, 3-52
 - name, specification of, 3-53
 - overview, 3-52
 - specification of, 3-53
 - use of, 1-14
- Screen definition errors, B-3
- SCREENTOARB, description, 7-19j
- Security display enhancement, 3-14, 3-33, 3-46, 3-49
- Selecting a function,

- FORMSPEC, 3-4
- REFSPEC, 5-14
- Separating data,
 - in forms, REFSPEC, 5-9
 - into several files, REFSPEC, 5-9, 5-12
- Separator, output field, REFSPEC, 5-26
- Sequence of USASCII characters, ascending order, C-1
- Sequence of forms,
 - altering in FORMSPEC, 4-50
 - ENTRY, 2-9
 - ENTRY, example, 2-10
 - REFSPEC input file, 5-5
- Session, terminating, 2-15
- SET statement, syntax, FORMSPEC, 4-37
- Set error flag, VPLUS/V intrinsic, 6-99
- Shift data to uppercase, FORMSPEC, 4-49
- SHOWCONTROL,
 - data capture devices, 6-96
 - definition, 6-16
 - local form storage, 6-16
 - VREADFIELDS, 6-96
 - VSHOWFORM, 6-108
- Sign,
 - in entered data, FORMSPEC, 3-15, 3-47
 - option, REFSPEC, 5-40
 - placement by REFSPEC, 5-40
- Simple collection, forms design, 1-6
- Single value comparisons, 4-23
- Source data entry, 1-1
- Source field, FORMSPEC SET statement, 4-40
- Specifications, processing,
 - correction of, 4-4
 - how to enter, 4-3
 - specifying entry, 2-5
- SPL,
 - interface to VPLUS/V overview, 1-14, 6-1
 - parameters, 6-8
 - sample program, A-1
- SPLIT MESSAGE PAUSE, 3-56
- SPLITPAUSE, definition of, 6-19
- Standard formatting, REFSPEC, 5-36
- Starting column,
 - input field, REFSPEC, 5-30
 - output field, REFSPEC, 5-31
- State code,
 - as constant value, 4-10
 - list of, F-1
- Statement summary, FORMSPEC editing, 4-8
- Statements, field processing,
 - summary, 4-7
 - syntax, 4-5
- Status, VPLUS/V procedures, 6-12
- STOP statement, FORMSPEC, 4-56
- Strapping terminal, G-5

- Stream REFORMAT program, 5-46
- STRIP,
 - option, REFSPEC, 5-38
 - statement, FORMSPEC, 4-46
 - syntax, 4-46
- Substring specification, REFSPEC, 5-31
- Supported terminal features, G-2
- Syntax,
 - FORMSPEC,
 - edit statements, 4-8
 - processing specifications, 4-5
- System constants,
 - output file, REFSPEC, 5-32
 - record terminator, REFSPEC, 5-25
- System crash, recovery, 2-15
- System errors, ENTRY, 2-14

T

- Table checks, FORMSPEC, 4-25
- Terminal,
 - buffer configuration, G-13
 - close, VPLUS/V intrinsic, 6-36
 - color enhancement, G-4
 - communication area, 6-9, G-11
 - data capture, G-4, G-10
 - edits, 4-16, G-3
 - features,
 - defining in FORMSPEC, 3-33
 - supported, 2-2, G-2
 - information, G-1
 - keyboard, 2-2
 - local form storage, 6-3
 - modified data tag, G-2
 - open, VPLUS/V intrinsic, 6-89
 - operator, using manual, 1-23
 - relabelling function keys, G-2
 - reset function, VOPENTERM, 6-78
 - security display enhancement, G-2
 - strapping, G-5
 - supported by VPLUS/V, 2-2, G-2,3,5
 - use, ENTRY, 2-2
 - X.25 operation, G-4,7,9
- Terminal/Language Selection Menu, 3-54
- Terminating,
 - ENTRY, 2-15
 - FORMSPEC, 3-7
 - REFSPEC, 5-13
- Terminator, output record, REFSPEC, 5-25
- errors, B-6.7
- TERMOPTIONS, definition of, 6-18
- Termtypes, choice of, G-2
- TESTLIST file, REFORMAT 5-45

Text fields, FORMSPEC, 3-9
 Timeouts, 6-19, 6-97, G-12
 Today's date, as constant value, 4-10
 Touch Feature, coding, E-7
 Transaction Processing, 1-1
 Transact/3000, VPLUS/V interface, 1-1
 TRANSACT, sample program, A-66
 Transfer data,
 from buffer, VPLUS/V intrinsic, 6-43
 to buffer, VPLUS/V intrinsic, 6-86
 Transparency operator, MATCH statement, 4-31

U

Underline,
 FORMSPEC code for, 3-14, 3-46
 VPLUS/V intrinsic code, 6-13
 Unexpected interruption,
 ENTRY, 2-15
 FORMSPEC, 3-7
 REFSPEC, 5-13
 Terminal recovery, G-14
 Unprotected fields, definition, 2-1, 2-11
 Uppercase,
 shift data to,
 FORMSPEC, 4-49
 REFSPEC, 5-25
 Upshift data, output file, REFSPEC, 5-25
 User-defined commands, 5-45
 User environment control file, G-14
 Usertime, 6-19
 Using REFSPEC, 5-13
 UPSHIFT statement, FORMSPEC, 4-49
 USASCII character set, C
 convert from, VPLUS/V intrinsic, 6-72
 convert to, VPLUS/V intrinsic, 6-101
 USBUFLEN, definition, 6-12

V

VPLUS/V, intrinsic messages, B-43
 error messages, B-1
 intrinsic,
 by function, 6-28
 call format, 6-7
 dependency between, 6-31
 error messages, list of, B-43
 flow, 6-29
 how to call, 6-7
 information, 6-48
 operational overview, 6-4
 overview, 6-1

- sample programs using, A-1
- status, COMAREA item, 6-12
- summary, 6-2
- use of, 1-14
- overview, 1-1, 6-1
- users, 1-14
- Value comparisons, FORMSPEC edits, 4-23
- VCHANGEFIELD, description of, 6-32
- VCLOSEBATCH, description, 6-34
- VCLOSEFORMF, definition, 6-35
- VCLOSETERM, description, 6-36
- VENVCNTL.PUB.SYS, G-14
- Verify check digit, FORMSPEC, 4-33
- VERRMSG, description, 6-42
- Version control, H-1
- VFIELDDEDITS, description, 6-39
- VFINISHFORM, description, 6-41
- VGETBUFFER, description, 6-43
- VGETFIELD, description, 6-45
- VGETFIELDINFO, description, 6-48
- VGETFILEINFO, description, 6-51
- VGETFORMINFO, description, 6-54
- VGETIEEEELONG, description, J-5
- VGETIEEEEREAL, description, J-3
- VGETKEYLABELS, description, 6-58
- VGETLANG, description, 6-60
- VGETNEXTFORM, description, 6-62
- VGETPACKED, description, 3-17a, 6-64
- VGETtype, description, 6-64
- VGETYYMMDD, description, 3-17a, 6-64
- VGETZONED, description, 3-17a, 6-64
- Viewing collected data, ENTRY, 2-17
- VINITFORM, description, 6-68
- VLOADFORMS, description, 6-70
- VOPENBATCH, description, 6-72
- VOPENFORMF, description, 6-76
- VOPENTERM, description, 6-78
- VPLACECURSOR, description, 6-80
- VPOSTBATCH, description, 6-82
- VPRINTFORM, description, 6-84
- VPRINTSCREEN, description, 6-85a
- VPUTBUFFER, description, 6-86
- VPUTFIELD, description, 6-88
- VPUTIEEEELONG, description, J-9
- VPUTIEEEEREAL, description, J-7
- VPUTPACKED, description, 6-90
- VPUTtype, description, 6-90
- VPUTWINDOW, description, 6-92
- VPUTYYMMDD, description, 6-90
- VPUTZONED, description, 6-90
- VREADBATCH, description, 6-94
- VREADFIELDS, description, 6-96
- VSETERROR, description, 6-99
- VSETKEYLABEL, description, 6-102

VSETKEYLABELS, description, 6-104
 VSETLANG, description, 6-106
 VSHOWFORM, description, 6-108
 VTURNOFF, description, G-16
 VTURNON, description, G-18
 VUNLOADFORM, description, 6-112
 VWRITEBATCH, description, 6-114

W

Window,
 area, 6-6
 enhancement, VPLUS/V code, 6-13
 line,
 enhancement, FORMSPEC, 3-49
 placement, FORMSPEC, 3-49
 move message to, VPLUS/V intrinsic, 6-92
 WINDOWENH, definition, 6-13
 Write data to,
 batch file, VPLUS/V intrinsic, 6-114
 buffer, VPLUS/V intrinsic, 6-86
 Write field and convert to USASCII, 6-90
 Write field to buffer, VPLUS/V intrinsic, 6-88

X

X.25 operation, G-4,7,9

Y

YMD field, FORMSPEC, 3-16, 3-47

\$

\$EMPTY, field processing constant, 4-10
 \$END, last form FORMSPEC, 3-41, 6-14
 \$HEAD, head form FORMSPEC, 3-41, 6-14
 \$LENGTH, field processing constant, 4-10
 \$REFRESH, refresh form FORMSPEC, 3-42, 6-14
 \$RETURN, next form FORMSPEC, 3-41, 6-14
 \$STATE, field processing constant, 4-10
 \$TODAY, field processing constant, 4-10

