WRDC-TR-90-8007
Volume VIII
Part 6

AD-A250 482

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 6 - Forms Processor User's Manual

S. Barker

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

DTIC
ELECTE
APR 14 1992
S B D

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

Approved for Public Release; Distribution is Unlimited

92-09483

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

92 4 13 071

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Unclassified

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UM 620344200	5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VIII, Part 6
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services 6b. OFFICE SYMBOL *(if applicable)*	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI
6c. ADDRESS *(City, State, and ZIP Code)* 2970 Presidential Drive Fairborn, OH 45324-6209	7b. ADDRESS *(City, State, and ZIP Code)* WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF 8b. OFFICE SYMBOL *(if applicable)* WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464
8c. ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB, Ohio 45433-6533	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. 78011F; PROJECT NO. 595600; TASK NO. F95600; WORK UNIT NO. 20950607
11. TITLE *(I* See block 19	
12. PERSONAL AUTHOR(S) Structural Dynamics Research Corporation: Barker, S., et al.	
13a. TYPE OF REPORT Final Report 13b. TIME COVERED 4/1/87-12/31/90	14. DATE OF REPORT *(Yr.,Mo.,Day)* 1990 September 30 15. PAGE COUNT 402
16. SUPPLEMENTARY NOTATION WRDC/MTI Project Priority 6203	

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify block no.)*
FIELD	GROUP	SUB GR.	
1308	0905		

19. ABSTRACT *(Continue on reverse if necessary and identify block number)*

This manual describes the Form Processor (FP) which is a set of callable execution time routines available to an application program for form processing.

BLOCK 11:

INTEGRATED INFORMATION SUPPORT SYSTEM
Vol VIII - User Interface Subsystem

Part 6 - Forms Processor User's Manual

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NO. *(Include Area Code)* (513) 255-7371 22c. OFFICE SYMBOL WRDC/MTI

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface, and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

Table of Contents

			Page
SECTION	1.0	INTRODUCTION	1-1
SECTION	2.0	DOCUMENTS	2-1
	2.1	Reference Documents	2-1
	2.2	Terms and Abbreviations	2-2
SECTION	3.0	FORM PROCESSING CONCEPTS	3-1
	3.1	Displaying Forms	3-1
	3.2	Building the Display List	3-1
	3.3	Opening Forms	3-2
	3.4	Identifying Window Contents	3-2
	3.5	Identifying Data on the Display List	3-3
	3.6	Application Structures	3-6
	3.6.1	Single Form Structure	3-7
	3.6.2	Simple Form Hierarchy	3-7
	3.6.3	Complex Form Hierarchy	3-8
	3.7	Error Handling	3-9
	3.8	Paging and Scrolling	3-10
	3.9	Logical Devices	3-11
	3.10	Creating and Modifying Forms at Run-time	3-11
SECTION	4.0	FP CALLABLE ROUTINES	4-1
	4.1	ADDDIM	4-4
	4.1.1	Calling Format	4-4
	4.1.2	Parameter Descriptions	4-4
	4.1.3	Example	4-5
	4.2	ADDELM	4-6
	4.2.1	Calling Format	4-6
	4.2.2	Parameter Descriptions	4-6
	4.2.3	Example	4-7
	4.3	ADDFLD	4-8
	4.3.1	Calling Format	4-8
	4.3.2	Parameter Descriptions	4-9
	4.3.3	Example	4-10
	4.4	ADDFRM	4-12
	4.4.1	Calling Format	4-12
	4.4.2	Parameter Descriptions	4-12
	4.4.3	Example	4-13
	4.5	APRFLD	4-14
	4.5.1	Calling Format	4-14
	4.5.2	Parameter Descriptions	4-14
	4.5.3	Example	4-15
	4.6	CHGLDV	4-16
	4.6.1	Calling Format	4-16
	4.6.2	Parameter Descriptions	4-16
	4.6.3	Example	4-17
	4.7	CLSFRM	4-18
	4.7.1	Calling Format	4-18
	4.7.2	Parameter Descriptions	4-18
	4.7.3	Example	4-18

Table of Contents (continued)

Page

4.8 CLSLDV 4-19
4.8.1 Calling Format 4-19
4.8.2 Parameter Descriptions 4-19
4.8.3 Example 4-19
4.9 CRTFLD 4-20
4.9.1 Calling Format 4-20
4.9.2 Parameter Descriptions 4-20
4.9.3 Example 4-21
4.10 CRTFRM 4-22
4.10.1 Calling Format 4-22
4.10.2 Parameter Descriptions 4-22
4.10.3 Example 4-22
4.11 GDATA 4-23
4.11.1 Calling Format 4-23
4.11.2 Parameter Descriptions 4-23
4.11.3 Example 4-24
4.12 GDATLN 4-25
4.12.1 Calling Format 4-25
4.12.2 Parameter Descriptions 4-25
4.12.3 Example 4-26
4.13 GDPFEX 4-27
4.13.1 Calling Format 4-27
4.13.2 Parameter Descriptions 4-27
4.13.3 Example 4-28
4.14 GDPFLC 4-29
4.14.1 Calling Format 4-29
4.14.2 Parameter Descriptions 4-29
4.14.3 Example 4-30
4.15 GETATT 4-31
4.15.1 Calling Format 4-31
4.15.2 Parameter Descriptions 4-31
4.15.3 Example 4-32
4.16 GETBAK 4-33
4.16.1 Calling Format 4-33
4.16.2 Parameter Descriptions 4-33
4.16.3 Example 4-34
4.17 GETCUR 4-35
4.17.1 Calling Format 4-35
4.17.2 Parameter Descriptions 4-35
4.18.3 Example 4-36
4.18 GETDQN 4-37
4.18.1 Calling Format 4-37
4.18.2 Parameter Descriptions 4-37
4.18.3 Example 4-37
4.19 GETLDV 4-38
4.19.1 Calling Format 4-38
4.19.2 Parameter Descriptions 4-38
4.19.3 Example 4-39

Table of Contents (continued)

Page

4.20 GETVTI 4-40
4.20.1 Calling Format 4-40
4.20.2 Parameter Descriptions 4-40
4.20.3 Example 4-41
4.21 GFMFLD 4-42
4.21.1 Calling Format 4-42
4.21.2 Parameter Descriptions 4-42
4.21.3 Example 4-43
4.22 GPAGE 4-44
4.22.1 Calling Format 4-44
4.22.2 Parameter Descriptions 4-44
4.22.3 Example 4-45
4.23 GTUINF 4-46
4.23.1 Calling Format 4-46
4.23.2 Parameter Descriptions 4-46
4.23.3 Example 4-47
4.24 GTUSYM 4-48
4.24.1 Calling Format 4-48
4.24.2 Parameter Descriptions 4-48
4.24.3 Example 4-48
4.25 GWINDO 4-49
4.25.1 Calling Format 4-49
4.25.2 Parameter Descriptions 4-49
4.25.3 Example 4-50
4.26 INITFP 4-51
4.26.1 Calling Format 4-51
4.26.2 Parameter Descriptions 4-51
4.26.3 Example 4-51
4.27 INITVT 4-52
4.27.1 Calling Format 4-52
4.27.2 Parameter Descriptions 4-52
4.27.3 Example 4-52
4.28 INQABS 4-53
4.28.1 Calling Format 4-53
4.28.2 Parameter Descriptions 4-53
4.28.3 Example 4-54
4.29 INQAPR 4-55
4.29.1 Calling Format 4-55
4.29.2 Parameter Descriptions 4-55
4.29.3 Example 4-56
4.30 INQATT 4-57
4.30.1 Calling Format 4-57
4.30.2 Parameter Descriptions 4-57
4.30.3 Example 4-58
4.31 INQDIM 4-60
4.31.1 Calling Format 4-60
4.31.2 Parameter Descriptions 4-60
4.31.3 Example 4-61

Table of Contents (continued)

Page

4.32 INQDIS 4-62
4.32.1 Calling Format 4-62
4.32.2 Parameter Descriptions 4-62
4.32.3 Example 4-63
4.33 INQDOM 4-64
4.33.1 Calling Format 4-64
4.33.2 Parameter Descriptions 4-64
4.33.3 Example 4-65
4.34 INQHLP 4-67
4.34.1 Calling Format 4-67
4.34.2 Parameter Descriptions 4-67
4.34.3 Example 4-68
4.35 INQLDV 4-69
4.35.1 Calling Format 4-69
4.35.2 Parameter Descriptions 4-69
4.35.3 Example 4-69
4.36 INQLOC 4-70
4.36.1 Calling Format 4-70
4.36.2 Parameter Descriptions 4-70
4.36.3 Example 4-72
4.37 INQPRO 4-73
4.37.1 Calling Format 4-73
4.37.2 Parameter Descriptions 4-74
4.37.3 Example 4-76
4.38 INQSIZ 4-77
4.38.1 Calling Format 4-77
4.38.2 Parameter Descriptions 4-77
4.38.3 Example 4-78
4.39 INQTYP 4-79
4.39.1 Calling Format 4-79
4.39.2 Parameter Descriptions 4-79
4.39.3 Example 4-80
4.40 INQVAL 4-81
4.40.1 Calling Format 4-81
4.40.2 Parameter Descriptions 4-81
4.40.3 Example 4-82
4.41 MAKFRM 4-83
4.41.1 Calling Format 4-83
4.41.2 Parameter Descriptions 4-83
4.41.3 Example 4-84
4.42 MOVLDV 4-85
4.42.1 Calling Format 4-85
4.42.2 Parameter Descriptions 4-85
4.42.3 Example 4-86
4.43 MVRFLD 4-87
4.43.1 Calling Format 4-87
4.43.2 Parameter Descriptions 4-87
4.43.3 Example 4-89

Table of Contents (continued)

Page

4.44 NUMELM 4-90
4.44.1 Calling Format 4-90
4.44.2 Parameter Descriptions 4-90
4.44.3 Example 4-90
4.45 OISCR 4-91
4.45.1 Calling Format 4-91
4.45.2 Parameter Descriptions 4-91
4.45.3 Example 4-92
4.46 OPNFRM 4-93
4.46.1 Calling Format 4-93
4.46.2 Parameter Descriptions 4-93
4.46.3 Example 4-94
4.47 OPNLDV 4-95
4.47.1 Calling Format 4-95
4.47.2 Parameter Descriptions 4-95
4.47.3 Example 4-96
4.48 OUTSCR 4-97
4.48.1 Calling Format 4-97
4.48.2 Parameter Descriptions 4-97
4.48.3 Example 4-97
4.49 PARFQN 4-98
4.49.1 Calling Format 4-98
4.49.2 Parameter Descriptions 4-98
4.49.3 Example 4-99
4.50 PDATA..... 4-100
4.50.1 CallingFormat..... 4-100
4.50.2 Parameter Descriptions 4-100
4.50.3 Example 4-101
4.51 PMSGLC 4-102
4.51.1 Calling Format 4-102
4.51.2 Parameter Descriptions 4-102
4.51.3 Example 4-102
4.52 PMSGLS 4-103
4.52.1 Calling Format 4-103
4.52.2 Parameter Descriptions 4-103
4.52.3 Example 4-103
4.53 PUTATT 4-104
4.53.1 Calling Format 4-104
4.53.2 Parameter Descriptions 4-104
4.53.3 Example 4-105
4.54 PUTBAK 4-106
4.54.1 Calling Format 4-106
4.54.2 Parameter Descriptions 4-106
4.54.3 Example 4-107
4.55 PUTCUR 4-108
4.55.1 Calling Format 4-108
4.55.2 Parameter Descriptions 4-108
4.55.3 Example 4-108

Table of Contents (continued)

Page

4.56 PUTLOC 4-109
4.56.1 Calling Format 4-109
4.56.2 Parameter Descriptions 4-109
4.56.3 Example 4-110
4.57 PUTVTI 4-111
4.57.1 Calling Format 4-111
4.57.2 Parameter Descriptions 4-111
4.57.3 Example 4-112
4.58 REPFLD 4-113
4.58.1 Calling Format 4-113
4.58.2 Parameter Descriptions 4-113
4.58.3 Example 4-114
4.59 REPFRM 4-115
4.59.1 Calling Format 4-115
4.59.2 Parameter Descriptions 4-115
4.59.3 Example 4-116
4.60 RMVAPR 4-117
4.60.1 Calling Format 4-117
4.60.2 Parameter Descriptions 4-117
4.60.3 Example 4-117
4.61 RMVARY 4-118
4.61.1 Calling Format 4-118
4.61.2 Parameter Descriptions 4-118
4.61.3 Example 4-119
4.62 RMVATT 4-120
4.62.1 Calling Format 4-120
4.62.2 Parameter Descriptions 4-120
4.62.3 Example 4-121
4.63 RMVDIM 4-122
4.63.1 Calling Format 4-122
4.63.2 Parameter Descriptions 4-122
4.63.3 Example 4-123
4.64 RMVDOM 4-124
4.64.1 Calling Format 4-124
4.64.2 Parameter Descriptions 4-124
4.64.3 Example 4-125
4.65 RMVFLD 4-126
4.65.1 Calling Format 4-126
4.65.2 Parameter Descriptions 4-126
4.65.3 Example 4-126
4.66 RMVHLP 4-127
4.66.1 Calling Format 4-127
4.66.2 Parameter Descriptions 4-127
4.66.3 Example 4-127
4.67 RMVPAG 4-128
4.67.1 Calling Format 4-128
4.67.2 Parameter Descriptions 4-128
4.67.3 Example 4-129

Table of Contents (continued)

Page

4.68 RMVPRO 4-130
4.68.1 Calling Format 4-130
4.68.2 Parameter Descriptions 4-130
4.68.3 Example 4-131
4.69 RMVVAL 4-132
4.69.1 Calling Format 4-132
4.69.2 Parameter Descriptions 4-132
4.69.3 Example 4-133
4.70 RPLFRM 4-134
4.70.1 Calling Format 4-134
4.70.2 Parameter Descriptions 4-134
4.70.3 Example 4-135
4.71 SAVFRM 4-136
4.71.1 Calling Format 4-136
4.71.2 Parameter Descriptions 4-136
4.71.3 Example 4-137
4.72 SETAPR 4-138
4.72.1 Calling Format 4-138
4.72.2 Parameter Descriptions 4-138
4.72.3 Example 4-139
4.73 SETATT 4-140
4.73.1 Calling Format 4-140
4.73.2 Parameter Descriptions 4-140
4.73.3 Example 4-142
4.74 SETDIM 4-143
4.74.1 Calling Format 4-143
4.74.2 Parameter Descriptions 4-143
4.74.3 Example 4-144
4.75 SETDIS 4-145
4.75.1 Calling Format 4-145
4.75.2 Parameter Descriptions 4-145
4.75.3 Example 4-146
4.76 SETDOM 4-147
4.76.1 Calling Format 4-147
4.76.2 Parameter Descriptions 4-147
4.76.3 Example 4-148
4.77 SETDQN 4-149
4.77.1 Calling Format 4-149
4.77.2 Parameter Descriptions 4-149
4.77.3 Example 4-149
4.78 SETHLP 4-150
4.78.1 Calling Format 4-150
4.78.2 Parameter Descriptions 4-150
4.78.3 Example 4-151
4.79 SETLDV 4-152
4.79.1 Calling Format 4-152
4.79.2 Parameter Descriptions 4-152
4.79.3 Example 4-153
4.80 SETLOC 4-154

Table of Contents (continued)

Page

4.80.1 Calling Format 4-154
4.80.2 Parameter Descriptions 4-154
4.80.3 Example 4-156
4.81 SETNAM 4-157
4.81.1 Calling Format 4-157
4.81.2 Parameter Descriptions 4-157
4.81.3 Example 4-158
4.82 SETPRO 4-159
4.82.1 Calling Format 4-159
4.82.2 Parameter Descriptions 4-159
4.82.3 Example 4-161
4.83 SETSIZ 4-162
4.83.1 Calling Format 4-162
4.83.2 Parameter Descriptions 4-162
4.83.3 Example 4-163
4.84 SETTYP 4-164
4.84.1 Calling Format 4-164
4.84.2 Parameter Descriptions 4-164
4.84.3 Example 4-165
4.85 SETVAL 4-166
4.85.1 Calling Format 4-166
4.85.2 Parameter Descriptions 4-166
4.85.3 Example 4-167
4.86 TERMFP 4-168
4.86.1 Calling Format 4-168
4.86.2 Parameter Descriptions 4-168
4.86.3 Example 4-168
4.87 TERMVT 4-169
4.87.1 Calling Format 4-169
4.87.2 Parameter Descriptions 4-169
4.87.3 Example 4-169

SECTION 5.0 INCLUDE FILES 5-1
5.1 FPPARM 5-1
5.2 FPCODE 5-3

SECTION 6.0 ACCESSING THE FP ROUTINES 6-1
6.1 Logicals 6-1
6.2 Linking 6-2

SECTION 7.0 SAMPLE APPLICATION PROGRAM 7-1

SECTION 8.0 LAYOUT OPTIMIZATION SYSTEM 8-1
8.1 Description 8-1
8.1.1 Getting Started 8-1

Table of Contents (continued)

Page

8.2 Interactive Chart Definition 8-6
8.2.1 Chart Functions 8-7
8.2.1.1 ADD Chart 8-8
8.2.1.2 UPDATE Chart 8-10
8.2.1.3 DELETE Chart 8-11
8.2.1.4 RETRIEVE Items 8-13
8.2.2 Object Functions 8-15
8.2.2.1 ADD Object 8-16
8.2.2.2 UPDATE Object 8-18
8.2.2.3 DELETE Object 8-19
8.2.2.4 RETRIEVE Items 8-21
8.2.3 Relation Functions 8-24
8.2.3.1 ADD Relation 8-25
8.2.3.2 UPDATE Relation 8-27
8.2.3.3 DELETE Relation 8-29
8.2.3.4 RETRIEVE Relation 8-31
8.3 Callable Routines 8-34
8.3.1 BEGCHT 8-36
8.3.1.1 Calling Format 8-36
8.3.1.2 Parameter Descriptions 8-36
8.3.1.3 Example 8-36
8.3.2 DCHART 8-37
8.3.2.1 Calling Format 8-37
8.3.2.2 Parameter Descriptions 8-37
8.3.2.3 Valid Keywords 8-38
8.3.2.4 Example 8-38
8.3.3 DELCHT 8-39
8.3.3.1 Calling Format 8-39
8.3.3.2 Parameter Descriptions 8-39
8.3.3.3 Example 8-39
8.3.4 DELOBJ 8-40
8.3.4.1 Calling Format 8-40
8.3.4.2 Parameter Descriptions 8-40
8.3.4.3 Example 8-41
8.3.5 DELREL 8-42
8.3.5.1 Calling Format 8-42
8.3.5.2 Parameter Descriptions 8-42
8.3.5.3 Example 8-43
8.3.6 DOBJTP 8-44
8.3.6.1 Calling Format 8-44
8.3.6.2 Parameter Descriptions 8-45
8.3.6.3 Valid Keywords 8-45
8.3.6.4 Example 8-46
8.3.7 DRELTP 8-47
8.3.7.1 Calling Format 8-47
8.3.7.2 Parameter Descriptions 8-48
8.3.7.3 Valid Keywords for Line-Style 8-49
8.3.7.4 Valid Keywords for Keyword-List 8-49
8.3.7.5 Example 8-50

Table of Contents (continued)

Page

8.3.8 DRWCHT 8-51
8.3.8.1 Calling Format 8-51
8.3.8.2 Parameter Descriptions 8-51
8.3.8.3 Example 8-51
8.3.9 ENDCHT 8-52
8.3.9.1 Calling Format 8-52
8.3.9.2 Parameter Descriptions 8-52
8.3.9.3 Example 8-52
8.3.10 GINSNM 8-53
8.3.10.1 Calling Format 8-53
8.3.10.2 Parameter Descriptions 8-53
8.3.10.3 Example 8-53
8.3.11 INITFL 8-54
8.3.11.1 Calling Format 8-54
8.3.11.2 Parameter Descriptions 8-54
8.3.11.3 Example 8-54
8.3.12 MAKCHT 8-55
8.3.12.1 Calling Format 8-55
8.3.12.2 Parameter Descriptions 8-55
8.3.12.3 Example 8-56
8.3.13 MAKOBJ 8-57
8.3.13.1 Calling Format 8-57
8.3.13.2 Parameter Descriptions 8-57
8.3.13.3 Example 8-58
8.3.14 MAKREL 8-59
8.3.14.1 Calling Format 8-59
8.3.14.2 Parameter Descriptions 8-60
8.3.14.3 Example 8-61
8.3.15 PRNCHT 8-62
8.3.15.1 Calling Format 8-62
8.3.15.2 Parameter Descriptions 8-62
8.3.15.3 Pagination Style Keywords 8-63
8.3.15.4 Example 8-63
8.3.16 TERMFL 8-64
8.3.16.1 Calling Format 8-64
8.3.16.2 Parameter Descriptions. 8-64
8.3.16.3 Example 8-64

SECTION 9.0 THREE DIMENSIONAL GRAPHICS 9-1
9.1 POPENPHIGS 9-3
9.1.1 Calling Sequence 9-3
9.1.2 Parameter Descriptions 9-3
9.1.3 Example 9-3
9.2 PCLOSEPHIGS 9-4
9.2.1 Calling Sequence 9-4
9.2.2 Parameter Descriptions 9-4
9.2.3 Example 9-4
9.3 POPENSTRUCT 9-5
9.3.1 Calling Sequence 9-5
9.3.2 Parameter Descriptions 9-5
9.3.3 Example 9-5

Table of Contents (continued)

		Page
9.4	PCLOSESTRUCT	9-6
9.4.1	Calling Sequence	9-6
9.4.2	Parameter Descriptions	9-6
9.4.3	Example	9-6
9.5	PEXECUTESTRUCT	9-7
9.5.1	Calling Sequence	9-7
9.5.2	Parameter Descriptions	9-7
9.5.3	Example	9-8
9.6	PDELSTRUCT	9-9
9.6.1	Calling Sequence	9-9
9.6.2	Parameter Descriptions	9-9
9.6.3	Example	9-9
9.7	PDELSTRUCTNET	9-10
9.7.1	Calling Sequence	9-10
9.7.2	Parameter Descriptions	9-10
9.7.3	Example	9-11
9.8	PDELALLSTRUCT	9-12
9.8.1	Calling Sequence	9-12
9.8.2	Parameter Descriptions	9-12
9.8.3	Example	9-12
9.9	PPOLYLINE3	9-13
9.9.1	Calling Sequence	9-13
9.9.2	Parameter Descriptions	9-13
9.9.3	Example	9-13
9.10	PPOLYLINE	9-14
9.10.1	Calling Sequence	9-14
9.10.2	Parameter Descriptions	9-14
9.10.3	Example	9-14
9.11	PPOLYMARKER3	9-15
9.11.1	Calling Sequence	9-15
9.11.2	Parameter Descriptions	9-15
9.11.3	Example	9-15
9.12	PPOLYMARKER	9-16
9.12.1	Calling Sequence	9-16
9.12.2	Parameter Descriptions	9-16
9.12.3	Example	9-16
9.13	PTEXT3	9-17
9.13.1	Calling Sequence	9-17
9.13.2	Parameter Descriptions	9-17
9.13.3	Example	9-18
9.14	PTEXT	9-19
9.14.1	Calling Sequence	9-19
9.14.2	Parameter Descriptions	9-19
9.14.3	Example	9-19
9.15	PFILLAREA3	9-20
9.15.1	Calling Sequence	9-20
9.15.2	Parameter Descriptions	9-20
9.15.3	Example	9-20

Table of Contents (continued)

Page

9.16 PFILLAREA 9-21
9.16.1 Calling Sequence 9-21
9.16.2 Parameter Descriptions 9-21
9.16.3 Example 9-21
9.17 POPENWS 9-22
9.17.1 Calling Sequence 9-22
9.17.2 Parameter Descriptions 9-22
9.17.3 Example 9-23
9.18 PCLOSEWS 9-24
9.18.1 Calling Sequence 9-24
9.18.2 Parameter Descriptions 9-24
9.18.3 Example 9-24
9.19 PPOSTSTRUCT 9-25
9.19.1 Calling Sequence 9-25
9.19.2 Parameter Descriptions 9-26
9.19.3 Example 9-26
9.20 PUNPOSTSTRUCT 9-27
9.20.1 Calling Sequence 9-27
9.20.2 Parameter Descriptions 9-27
9.20.3 Example 9-27
9.21 PUNPOSTALLSTRUCT 9-29
9.21.1 Calling Sequence 9-29
9.21.2 Parameter Descriptions 9-29
9.21.3 Example 9-29
9.22 PMSG 9-31
9.22.1 Calling Sequence 9-31
9.22.2 Parameter Descriptions 9-31
9.22.3 Example 9-32
9.23 PSETLINETYPE 9-33
9.23.1 Calling Sequence 9-33
9.23.2 Parameter Descriptions 9-33
9.23.3 Example 9-34
9.24 PSETLINECOLOURIND 9-35
9.24.1 Calling Sequence 9-35
9.24.2 Parameter Descriptions 9-35
9.24.3 Example 9-36
9.25 PSETMARKERTYPE 9-37
9.25.1 Calling Sequence 9-37
9.25.2 Parameter Descriptions 9-37
9.25.3 Example 9-38
9.26 PSETMARKERCOLOURIND 9-39
9.26.1 Calling Sequence 9-39
9.26.2 Parameter Descriptions 9-39
9.26.3 Example 9-40
9.27 PSETTEXTCOLOURIND 9-41
9.27.1 Calling Sequence 9-41
9.27.2 Parameter Descriptions 9-41
9.27.3 Example 9-41

Table of Contents (continued)

Page

9.28 PSETCHARHEIGHT 9-43
9.28.1 Calling Sequence 9-43
9.28.2 Parameter Descriptions 9-43
9.28.3 Example 9-43
9.29 PSETINTSTYLE 9-45
9.29.1 Calling Sequence 9-45
9.29.2 Parameter Descriptions 9-45
9.29.3 Example 9-45
9.30 PTRANSLATE3 9-47
9.30.1 Calling Sequence 9-47
9.30.2 Parameter Descriptions 9-47
9.30.3 Example 9-48
9.31 PTRANSLATE 9-49
9.31.1 Calling Sequence 9-49
9.31.2 Parameter Descriptions 9-49
9.31.3 Example 9-50
9.32 PSCALE3 9-51
9.32.1 Calling Sequence 9-51
9.32.2 Parameter Descriptions 9-51
9.32.3 Example 9-52
9.33 PSCALE 9-53
9.33.1 Calling Sequence 9-53
9.33.2 Parameter Descriptions 9-53
9.33.3 Example 9-54
9.34 PROTATEX 9-55
9.34.1 Calling Sequence 9-55
9.34.2 Parameter Descriptions 9-55
9.34.3 Example 9-56
9.35 PROTATEY 9-57
9.35.1 Calling Sequence 9-57
9.35.2 Parameter Descriptions 9-57
9.35.3 Example 9-58
9.36 PROTATEZ 9-59
9.36.1 Calling Sequence 9-59
9.36.2 Parameter Descriptions 9-59
9.36.3 Example 9-60
9.37 PROTATE 9-61
9.37.1 Calling Sequence 9-61
9.37.2 Parameter Descriptions 9-61
9.37.3 Example 9-62
9.38 PCOMPOSEMATRIX3 9-63
9.38.1 Calling Sequence 9-63
9.38.2 Parameter Descriptions 9-63
9.38.3 Example 9-64
9.39 PCOMPOSEMATRIX 9-66
9.39.1 Calling Sequence 9-66
9.39.2 Parameter Descriptions 9-66
9.39.3 Example 9-67

Table of Contents (continued)

Page

9.40 PTRANPT3 9-69
9.40.1 Calling Sequence 9-69
9.40.2 Parameter Descriptions 9-69
9.40.3 Example 9-70
9.41 PTRANPT 9-71
9.41.1 Calling Sequence 9-71
9.41.2 Parameter Descriptions 9-71
9.41.3 Example 9-71
9.42 PBUILDTRAN3 9-73
9.42.1 Calling Sequence 9-73
9.42.2 Parameter Descriptions 9-73
9.42.3 Example 9-74
9.43 PBUILDTRAN 9-76
9.43.1 Calling Sequence 9-76
9.43.2 Parameter Descriptions 9-76
9.43.3 Example 9-77
9.44 PCOMPOSETRAN3 9-78
9.44.1 Calling Sequence 9-78
9.44.2 Parameter Descriptions 9-78
9.44.3 Example 9-79
9.45 PCOMPOSETRAN 9-81
9.45.1 Calling Sequence 9-81
9.45.2 Parameter Descriptions 9-81
9.45.3 Example 9-82
9.46 PSETINTCOLOURIND 9-83
9.46.1 Calling Sequence 9-83
9.46.2 Parameter Descriptions 9-83
9.46.3 Example 9-83
9.47 PSETLOCALTRAN3 9-85
9.47.1 Calling Sequence 9-85
9.47.2 Parameter Descriptions 9-85
9.47.3 Example 9-86
9.48 PSETLOCALTRAN 9-87
9.48.1 Calling Sequence 9-87
9.48.2 Parameter Descriptions 9-87
9.48.3 Example 9-88
9.49 PSETGLOBALTRAN3 9-89
9.49.1 Calling Sequence 9-89
9.49.2 Parameter Descriptions 9-89
9.49.3 Example 9-89
9.50 PSETGLOBALTRAN 9-91
9.50.1 Calling Sequence 9-91
9.50.2 Parameter Descriptions 9-91
9.50.3 Example 9-91
9.51 PSETWSWINDOW3 9-93
9.51.1 Calling Sequence 9-93
9.51.2 Parameter Descriptions 9-93
9.51.3 Example 9-96

Table of Contents (continued)

Page

9.52 PSETWSWINDOW 9-97
9.52.1 Calling Sequence 9-97
9.52.2 Parameter Descriptions 9-97
9.52.3 Example 9-100
9.53 PSETWSVIEWPORT3 9-101
9.53.1 Calling Sequence 9-101
9.53.2 Parameter Descriptions 9-101
9.53.3 Example 9-102
9.54 PSETWSVIEWPORT 9-103
9.54.1 Calling Sequence 9-103
9.54.2 Parameter Descriptions 9-103
9.54.3 Example 9-104
9.55 PREDRAWALLSTRUCT 9-105
9.55.1 Calling Sequence 9-105
9.55.2 Parameter Descriptions 9-105
9.55.3 Example 9-105
9.56 PUPDATEWS 9-107
9.56.1 Calling Sequence 9-107
9.56.2 Parameter Descriptions 9-107
9.56.3 Example 9-107
9.57 PINQWSTYPES 9-109
9.57.1 Calling Sequence 9-109
9.57.2 Parameter Descriptions 9-109
9.57.3 Example 9-109
9.58 PINQWSCONNTYPE 9-110
9.58.1 Calling Sequence 9-110
9.58.2 Parameter Descriptions 9-110
9.58.3 Example 9-110
9.59 PINQDISPLAYSPACESIZE3 9-112
9.59.1 Calling Sequence 9-112
9.59.2 Parameter Descriptions 9-112
9.59.3 Example 9-113
9.60 PINQDISPLAYSPACESIZE 9-114
9.60.1 Calling Sequence 9-114
9.60.2 Parameter Descriptions 9-114
9.60.3 Example 9-114
9.61 PHIGS Data Structures 9-115

LIST OF ILLUSTRATIONS

Figure	Title	Page
3-1	Display List Hierarchy	3-1
3-2	After Two ADDFRM Calls	3-3
3-3	After RPLFRM	3-3
3-4	After RMVPAG	3-3
3-5	Form Hierarchy	3-5
3-6	Sample Display List	3-6
3-7	Module Hierarchy	3-9
3-8	Message Management Scenario for Error Handling	3-10
7-1	Sample Form	7-1
8-1	Sample Chart	8-2
8-2	Sample Object Archetype Definition Form	8-3
8-3	Sample Termination Symbol Archetype Definition Form	8-4
8-4	Figure 8-4	8-6
8-5	Figure 8-5	8-7
8-6	Figure 8-6	8-8
8-7	Figure 8-7	8-9
8-8	Figure 8-8	8-10
8-9	Figure 8-9	8-11
8-10	Figure 8-10	8-11
8-11	Figure 8-11	8-13
8-12	Figure 8-12	8-14
8-13	Figure 8-13	8-15
8-14	Figure 8-14	8-16
8-15	Figure 8-15	8-17
8-16	Figure 8-16	8-18
8-17	Figure 8-17	8-19
8-18	Figure 8-18	8-20
8-19	Figure 8-19	8-21
8-20	Figure 8-20	8-22
8-21	Figure 8-21	8-23
8-22	Figure 8-22	8-24
8-23	Figure 8-23	8-26
8-24	Figure 8-24	8-27
8-25	Figure 8-25	8-28
8-26	Figure 8-26	8-29
8-27	Figure 8-27	8-30
8-28	Figure 8-28	8-31
8-29	Figure 8-29	8-32
8-30	Figure 8-30	8-33

LIST OF TABLES

TABLE	TITLE	PAGE
8.1	Callable Layout System Routines.....	8
8.1	Example Data Types.....	8-34

SECTION 1

INTRODUCTION

This manual describes the Form Processor (FP) which is a set of callable execution time routines available to an application program for form processing. The information is addressed to application programmers using the Integrated Information Support System (IISS).

The FP routines allow programs and their users to communicate through pre-defined forms on a terminal. The forms can be defined at run-time using FP routines or by using the Form Editor. When the application runs, the FP handles:

- Retrieval of a pre-defined form
- Display of the form
- Filling out of the form (from the user point of view)
- Data integrity checking
- Providing a help facility
- Definition of new forms at run-time
- Modification of existing forms at run-time

The application program can be written in most of the high level programming languages. However, since the form data are highly structured, it is wise to choose a language that supports data structures such as PL1, COBOL, or C.

All FP calls refer to specific forms and data within forms by names that are assigned during form definition. A typical order of events in an application program using these routines is:

1. Initialize Form Processor.
2. Open forms.
3. Add forms to windows.
4. Put data in forms.
5. Put data in message line.
6. Send screen to terminal and get input from the user.
7. Read data from forms.

When processing has finished, close forms and go to #10.

8. Remove pages that will not be displayed or replaced.
9. Replace forms if necessary.

To continue processing, return to #3.

10. Terminate Form Processor.

The concepts of form processing as well as terms such as windows and pages are explained in Section 3 of this manual. Details of the callable FP routines are given in Section 4.

SECTION 2

DOCUMENTS

2.1 Reference Documents

[1] Structural Dynamics Research Corporation, IISS Form Processor Development Specification, DS 620244200A, 16 February 1987.

[2] Systran, ICAM Documentation Standards, IDS150120000C, 15 September 1983.

This manual is one of a set of user's manuals that together describe how to operate in the IISS environment. The complete set consists of the following manuals listed here for reference:

[1] Structural Dynamics Research Corporation, IISS Form Editor User's Manual, UM 620244400A, 16 February 1987.

Explains how to define and maintain electronic forms. This manual is intended for use by programmers writing application programs that use the Form Processor.

[2] Structural Dynamics Research Corporation, IISS Form Processor User's Manual, UM 620244200A, 16 February 1987.

Describes the set of callable, execution-time routines available to an application program for the processing of electronic forms. This manual is intended for use by programmers writing application programs for the IISS environment.

[3] Structural Dynamics Research Corporation, IISS Terminal Operator Guide, OM 620244000A, 16 February 1987.

Explains how to operate the generic IISS terminal when running an IISS application program. The IISS end user environment, function selection, and some pre-defined applications are also described.

[4] Structural Dynamics Research Corporation, IISS Text Editor User's Manual, UM 620244600A, 16 February 1987.

Explains the use of file editing functions including: insert, delete, move, and replace text.

[5] Structural Dynamics Research Corporation, IISS Rapid Application Generator User's Manual, UM 620244502A, 16 February 1987.

Describes the Application Definition Language (ADL) subset of Form Definition Language (FDL), and outlines the process used for translating textual definitions of interactive database applications into programs that access selected data base

information resident in the Common Data Model (CDM). This information is accessible through the IISS Neutral Data Manipulation Language (NDML).

[6] Structural Dynamics Research Corporation, IISS Report Writer User's Manual, UM 620244501A, 16 February 1987.

Describes the Report Definition Language (RDL) subset of Form Definition Language (FDL), and outlines the process of creating a hard copy report of selected data base information resident in the Common Data Model (CDM). This information is accessible through the IISS Neutral Data Manipulation Language (NDML).

[7] Structural Dynamics Research Corporation, IISS Virtual Terminal User's Manual, UM 620244300A, 16 February 1987.

Explains the program callable interface to the IISS Virtual Terminal. The callable routines, Virtual Terminal commands, and the implementation of additional terminals are described. This manual is intended for use by application and system programmers working in the IISS environment.

2.2 Terms and Abbreviations

American Standard Code for Information Interchange (ASCII): The character set defined by ANSI X3.4 and used by most computer vendors.

Application Interface (AI): A subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The AI enables applications to be hosted on computers other than the host of the User Interface.

Application Process (AP): A cohesive unit of software that can be initiated as a unit to perform some function or functions.

Attribute: A field characteristic such as blinking, highlighted, black, etc. and various other combinations. Background attributes are defined for forms or windows only. Foreground attributes are defined for items. Attributes may be permanent, i.e., they remain the same unless changed by the application program, or they may be temporary, i.e., they remain in effect until the window is redisplayed.

Child Structure: A structure that is referenced from within another structure (the "parent structure") via an Execute Structure Element.

Common Data Model (CDM): The IISS subsystem that describes common data application process formats, form definitions, etc. of the IISS and includes conceptual schema, external schemas, internal schemas, and schema transformation operators.

Computer Program Configuration Item (CPCI): An aggregation of computer programs or any of their discrete portions, which satisfies an end-use function.

Conceptual Schema (CS): The standard definition used for all data in the CDM. It is based on IDEF1 information modeling.

Current Cursor Position: The position of the cursor before an edit command or function is issued in the text editor.

Cursor Position: The position of the cursor after any command is issued.

Device Drivers (DD): Software modules written to handle I/O for a specific kind of terminal. The modules map terminal specific commands and data to a neutral format. Device Drivers are part of the UI Virtual Terminal.

Display List: A list of all the open forms that are currently being processed by the FP or the user.

Display Size: The number of lines used in the edit area.

Execute Structure Element: An instruction within a structure (the "parent" structure) which points to another structure (the "child" structure). During traversal of the parent structure, when an execute structure element is encountered, traversal of the parent structure is stopped and the child structure is traversed. When traversal of the child structure has completed, traversal of the parent structure resumes at the point where the execute structure element was encountered.

Extended Binary Coded Decimal Interchange Code (EBCDIC): The character set used by a few computer vendors (notably IBM) instead of ASCII.

External Schema (ES): An application's view of the CDM's conceptual schema.

Field: Two dimensional space on a terminal screen.

Field Pointer: Indicates the ITEM which contains the current cursor position.

Form: Structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, and windows.

Form Definition (FD): Form Definition Language after compilation. It is read at run-time by the Form Processor.

Form Definition Language (FDL): The language in which electronic forms are defined.

Form Driven Form Editor (FDFE): The subset of the Form Editor which consists of a form driven application used to create Form Definition files interactively.

Form Editor (FE): A subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Form Driven Form Editor and the Form Language Compiler.

Form Hierarchy: A graphic representation of the way in which forms, items and windows are related to their parent form.

Form Language Compiler (FLAN): A subset of the Form Editor which translates Form Definition Language statements into Form Definition files.

Form Processor (FP): A subset of the IISS User Interface that consists of a set of callable, execution-time routines available to an application program for form processing.

Form Processor Text Editor (FPTE): A subset of the Form Processor that consists of software modules that provide text editing capabilities to all users of applications that use the Form Processor.

Integrated Information Support System (IISS): A test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network.

Item: A non-decomposable area of a form into which hard-coded descriptive text may be placed. An item is the only defined area where user data may be input/output.

Logical Device: A conceptual device that identifies a top level window of an application. It is used to distinguish between multiple applications running simultaneously on a physical device. NOTE that a single application can have more than one logical device. To the end user this also appears as multiple applications running simultaneously.

Message: Descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or provide other user information.

Message Line: A line on the terminal screen that is used to display messages.

Network Transaction Manager (NTM): The IISS subsystem that performs the coordination, communication and housekeeping functions required to integrate the Application Processes and System Services resident on the various hosts into a cohesive system.

Open List: A list of all the forms that are currently open for an application process.

Operating System (OS): Software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Page: An instance of a form in a window that is created whenever a form is added to a window.

Paging and Scrolling: A method which allows a form to contain more data than can be displayed with provisions for viewing any portion of the data buffer.

Parent Structure: A structure which references another structure (the "child" structure) via an Execute Structure Element.

Physical Device: A hardware terminal.

Presentation Schema (PS): May be equivalent to a form. It is the view presented to the user of the application.

Previous Cursor Position: The position of the cursor when the previous edit command was issued.

Qualified Name: The name of a form, item or window preceded by the hierarchy path so that it is uniquely identified.

State List: A list of conditions which are in effect for the currently open structure, e.g., current polyline attributes are the color blue, line width 1, line type solid, open workstation is number 1. These attributes stay in effect until another structure is opened which changes the state list.

Structure: A list of instructions that, when executed, will produce a graphics picture.

Structure Element: One of the instructions in the structure which is used to create a graphics picture.

Subform: A form that is used within another form.

Transformation: The process of changing the size and/or orientation of a displayed graphics picture. The transformations are made via the use of coordinate geometry, trigonometry, and matrix mathematics.

Traversal: The process of executing the list of instructions in a structure in order to create a graphics picture.

User Data: Data which is either input by the user or output by the application programs to items.

User Interface (UI): The IISS subsystem that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System (UIDS): The collection of IISS User Interface subsystems that are used by applications programmers as they develop IISS applications. The UIDS includes the Form Editor and the Application Generator.

User Interface Management System (UIMS): The run-time UI. The UIMS consists of the Form Processor, Virtual Terminal, Application Interface, the User Interface Services and the Text Editor.

User Interface Monitor (UIM): The part of the Form Processor that handles messaging between the NTM and the UI. The UIM also provides authorization checks and initiates applications.

User Interface/Virtual Terminal Interface (UI/VTI): Another name for the User Interface.

Virtual Terminal (VT): A subset of the IISS User Interface that performs the interfacing between different terminals and the UI. This is done by defining a specific set of terminal features and protocols which must be supported by the UI software which constitutes the virtual terminal definition. Specific terminals are then mapped against the virtual terminal software by specific software modules written for each type of real terminal supported.

Virtual Terminal Interface (VTI): The callable interface to the VT.

Window: A Dynamic area of a terminal screen on which pre-defined forms may be placed at run-time.

Window Manager: The facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, and the position of a form within a window. The Window Manager is part of the Form Processor.

SECTION 3

FORM PROCESSING CONCEPTS

This section explains several FP implementation concepts that will enable you to use the callable routines effectively.

3.1 Displaying Forms

The terminal screen can be thought of as a window in which forms are displayed. To determine what the user of your application will see in this window, your application must build a Display List. As explained in the Form Editor User's Manual, a form is composed of fields which can be defined as subforms, items, graphs, graphics primitives, and windows. Since windows can contain forms, which in turn can contain forms, items, graphs, graphics primitives (polylines, polymarkers, fill areas, and graphics text), and windows, the Display List can be a form hierarchy like that shown in Figure 3-1.

Figure 3-1 Display List Hierarchy

3.2 Building the Display List

The Display List must be built from the top down. At the top of the Display List will always be the form PSCREN, which is added to the Display List as part of initialization. PSCREN contains the window SCREEN which is where top-level user forms are placed. The include file FPPARM defines SCREN which refers to this window. A form cannot be added to a window unless that window is already

on the Display List. The Display List is formed as the application dynamically adds, removes, and replaces forms in windows using the routines ADDFRM, RPLFRM, and RMVPAG.

3.3 Opening Forms

In order to be put on the Display List or to be modified at run-time, a form must be brought into memory, or "opened". This is done either explicitly when OPNFRM is called, or implicitly when ADDFRM is called to actually add the form to the Display List. There are several factors to consider when deciding which type of open is more appropriate for each situation.

When the application explicitly opens all forms, the following will be true:

- Startup is slow
- All necessary forms are available
- Missing form errors are avoided
- Response time is consistent
- Resources may be wasted opening and closing forms that are not used

When the application implicitly opens forms, the following conditions are true:

- Startup is fast
- Missing form errors can occur
- Response time is inconsistent - first access takes longer since the form must be opened
- Resources are not wasted

3.4 Identifying Window Contents

As the application builds the Display List, forms may become layered in a window, i.e., they become stacked on top of each other, like pages in a book. Page one may obscure page two from sight, but page two becomes visible when page one is moved out of the way. Within the Form Processor, a page is an instance of a form in a window. Adding a form to a window causes a page to be created. Removing a form from the window deletes a page. The form on any given page may also be replaced by a different form.

The following figures illustrate how pages are created as forms are added, removed, and replaced in a window. The upper left hand corners of the boxes contain the form name. The upper right hand corners contain the page number.

In the first example, FORM1 has been added to the window using ADDFRM, creating page 1. Next, page 2 has been created by calling ADDFRM to add FORM2 to the window.

```
        +-----------------+
        |FORM1          1|
        |                 |
+-------+-----------+     |
|FORM2          2|        |
|                 +-------+
|                 |
|                 |
+-----------------+
```

Figure 3-2 After Two ADDFRM Calls

In Figure 3-3, page 2 has been altered by replacing FORM2 with FORM3 using RPLFRM.

```
        +-----------------+
        |FORM1          1|
        |                 |
+-------+-----------+     |
|FORM3          2|        |
|                 +-------+
|                 |
|                 |
+-----------------+
```

Figure 3-3 After RPLFRM

In Figure 3-4, RMVPAG has been used to remove page 2 from the window so that only page 1 remains.

```
+-----------------+
|FORM1          1|
|                 |
|                 |
|                 |
+-----------------+
```

Figure 3-4 After RMVPAG

These calls are used to build the Display List. To display the objects on the Display List on the terminal, the routines OUTSCR or OISCR must be called.

3.5 Identifying Data on the Display List

As illustrated in Figure 3-1, it is possible to have a form that is used in more than one window or as a subform of more than one form. When forms are used in this manner, it is necessary to have a means of identifying the specific Display List element in which to place data or from which to retrieve data. A similar

problem exists when a form contains arrays (i.e., repeating occurrences of a field). To solve these problems, Display List elements are identified using qualified names.

A fully qualified name specifies all forms, windows, and items on a path to the Display List element being identified with page numbers specified for all forms within windows and subscripts specified for all array elements. If a qualified name begins with a dot ("."), it is an absolute name and the path starts at the top of the Display List (i.e., the form PSCREN). A qualified name which does not begin with a dot is a relative name that starts at a default path on the Display List. The default path is set according to the most recent call to the routine SETDQN. By using relative names and SETDQN, a procedure can be written which uses a form without any knowledge of the window(s) in which the form is currently displayed.

To identify a specific Display List element, its qualified name need only include enough elements in the path so that the path is unique. A qualified name has the general form:

name1.name2. ... namen;

If there are repeating occurrences of a field in the path, a specific occurrence following the field name may optionally be specified. If an occurrence is not specified, all of the occurrences are searched or used depending upon whether the array field is along the path or at the end of it. If the path includes a window, the name may also be followed by an optional page number. The routines GWINDO and GPAGE are used to find out the number of pages in a window and the name of the form on a specific page, respectively. If the page number is not specified, the current page is used. The entire qualified name is followed with a semicolon (';').

The Backus-Naur specification for qualified names is:

<qualified name> ::=[.]<subname> [<subname_list>];

<subname_list> ::= .<subname> [<subname_list>]

<subname> ::= <name> [(<subscript_list>)] ["<"<page#>">"]

<subscript_list> ::= <subscript#> [,<subscript_list>]

A <name> may consist of up to ten alphanumeric and underscore ('_') characters. A form name may not contain underscores, however and specific systems may further restrict the length. A <name> must begin with an alphabetic character and cannot contain blanks. For example, item_one, field2, and ARRAY are all valid names. The names "7item", "item 2", and "help_form" are all illegal names.

Entries enclosed in square brackets ('[ ]') are optional.

Page number and subscript number may be absolute or relative as explained previously in this section, and cannot contain leading spaces. Leading zeros are allowed. For example, "1" and "01" both refer to the first page in a window.

To determine if a qualified name is legal, the following rules are applied:

1. The qualified name specified must identify a path in the current form hierarchy of the application. To identify a path, the fields in the name must exist, and they must be in the correct hierarchical order. Fields along the path may be omitted as long as the fields included are in the correct order. An error is returned if the path does not exist.

2. If the qualified name identifies more than one Display List element, the field is chosen whose first qualifier occurs highest in the hierarchy.

3. If rule 2 does not resolve the conflict, the path is chosen which spans the fewest levels in the hierarchy.

4. If rule 3 does not resolve the conflict, an error is returned.

```
A              Level 1
|
B              Level 2
|
+--+--+
|     |
C     D        Level 3
      |
   +--+--+
   |     |
   E     B     Level 4
         |
         C     Level 5
```

Figure 3-5 Form Hierarchy

Using the form hierarchy shown in Figure 3-5, the name "B.C" could refer to C at Level 3 or Level 5. When Rule 2 is applied, we see that C at Level 3 would be used. The name "A.C" could also be used to identify either C. When Rule 3 is applied, we again see that C at Level 3 would be used. If C also branched from E, there would be two C's at the same level. Then, if the path name was "D.C", an error would result. To access the C's at Level 5, you would have to specify a qualified name of "D.B.C" or "D.E.C".

In Figure 3-6, the window SCREEN on form PSCREN contains the form FORM1. FORM1 contains two windows, WINDOW1 and WINDOW2, and seven items, I1, I2, I3, and four occurrences of I4. Both WINDOW1 and WINDOW2 contain FORM2 which contains items I1 and I2.

```
+-----------------------------------------------+
|PSCREN                                         |
   +----SCREEN-------------------------------+
   |                                         |
   |FORM1                                    |
                          +WINDOW1----------------+
    I1     I2   I3   I4   |FORM2                  |
    +WINDOW2-----+  I4    | I1                    |
    |FORM2       |  I4    | I2                    |
    |I1          |  I4    |                       |
    |I2          |        |                       |
    |            |        |                       |
+---+-+----------+----+----------------------+
```

Figure 3-6 Sample Display List

Assuming that SETDQN was called with the qualified name ".WINDOW1;", some qualified names you might use to identify elements in this Display List are:

.PSCREN.SCREEN<0>.FORM1.WINDOW1<0>.FORM2.I2; - This is the fully qualified name of I2 on the current page of WINDOW1, which is on the current page of SCREEN.

.PSCREN.SCREEN<99>.FORM1.WINDOW1<98>.FORM2.I2; - This is the fully qualified name of I2 on page 98 of WINDOW1, which is on page 99 of SCREEN.

.FORM1.I2; - This identifies the item I2 on FORM1.

.FORM1; - This identifies the form FORM1.

.I3; - This identifies the item I3.

.WINDOW1; - This identifies the window WINDOW1.

.I4(3); - This identifies the third occurrence of the item I4.

I2; - This identifies the item I2 on FORM2 in WINDOW1.

Some illegal names are:

I2.FORM2; because the fields are specified in the wrong order.

.FORM2; because FORM2 is contained in two windows that are at the same level in the hierarchy.

3.6 Application Structures

The application structure should reflect the transaction oriented nature of the Form Processor and follow the structure of the application forms. An application can have three possible form structures:

- o Single form
- o Simple form hierarchy
- o Complex form hierarchy or network

3.6.1 Single Form Structure

An application that uses a single form can do all of its form processing in a "do forever" loop. The skeleton structure for an application that uses a single form is:

```
CALL INITAL
CALL INITFP
CALL OPNFRM
CALL ADDFRM
initialize form
DO FOREVER
     {
     CALL PDATA
     CALL OISCR
     IF PFKEY = QUITKEY THEN EXIT LOOP
     CALL GDATA
     process function key
     }
CALL CLSFRM
CALL TERMFP
CALL TRMNAT (or TRMNDML)
```

3.6.2 Simple Form Hierarchy

In an application that uses a simple form hierarchy, all forms are added to the window SCREN and you only go up and down the hierarchy. The main module in this application is similar to the single form application except for the do forever loop. With a simple form hierarchy, there is one module for each form and the main module calls the first form module. Each form module then adds its form, initializes it, processes it, and either calls the module for a form in the next level of the hierarchy or returns to the module from the previous level. The skeleton structure for a form module is:

```
CALL ADDFRM
initialize form
DO FOREVER
  {
  CALL PDATA
  CALL OISCR
  IF PFKEY = QUITKEY THEN EXIT LOOP
    {
    CALL GDATA
    process function key
    }
  IF return_to_previous_form THEN EXIT LOOP
  CALL next_form_module
  }
CALL RMVPAG
RETURN
```

3.6.3 Complex Form Hierarchy

In an application that uses a complex form hierarchy or network of forms, forms can be added to any window in the hierarchy and you can go across the hierarchy as well as up and down it. The main module in this application is a transaction dispatcher that does the call to OISCR and interprets the function keys. There is still one module for each form and the main module tells the form routine whether to initialize, process, or terminate. The skeleton structure for the main module is:

```
CALL INITAL
CALL INITFP
CALL first_form_routine(INIT)
DO FOREVER
{
CALL OISCR
CALL GWINDO(window, page)
CALL GPAGE(window, page, current_form)
IF PFKEY = QUITKEY THEN
  {
  CALL current_form_routine(TERM)
  IF page = 1 THEN EXIT LOOP
  }
CALL current_form_routine(PROC)
IF next_form != current_form THEN
  {
  IF NOT nested THEN CALL current_form_routine(TERM)
  ELSE CALL next_form_routine(INIT)
  }
}
```

The skeleton structure for a form module is:

```
SWITCH (action)
  {
  CASE INIT:
    CALL ADDFRM
    initialize form
    CALL PDATA
  CASE PROC:
    CALL GDATA
    process form data
    CALL PDATA
  CASE TERM:
    CALL RMVPAG
  }
RETURN
```

Figure 3-7 shows the appearance of the module hierarchy for this type of application:

Figure 3-7 Module Hierarchy

3.7 Error Handling

The FP supports two ways of indicating to the application user that an error has occurred. The routine PUTATT can be used to highlight item fields in error by temporarily changing their display attributes and error messages can be put in the last line of the form PSCREN by the Form Processor or the application program. When creating forms, the last line of the physical terminal screen cannot contain any form data.

To write to this line from your application, call one of the two routines that store the message line (PMSGLC or PMSGLS) and then output the form to the terminal using OUTSCR or OISCR. PMSGLS places a character string in the message line. PMSGLC accepts a code which corresponds to a message string that resides in a file. This file is pre-defined and can be created and maintained using the User Interface tool Message Management. INCGEN is another UI tool that is then used to generate include files containing message code declarations. Figure 3-8 illustrates this type of error handling.

```
+----------+  Messages  +-------+ Message codes  +----------+
| Message  |<---------->|Message+---------------->|  INCGEN  |
|Management| and  codes |  File |  & data names  |          |
+----------+            |       |                +-----+----+
                        +---+---+             Message |
                            |                    Code |
                            |                Declarations|
                            |                         v
                            |                   +-------+
                            +---+               |Message|
                                |               | Code  |
                                |               |Include|
                                |               +---+---+
                                v                   |
                                                    v
+--------+  Message +----------+  Message +-----------+
|Terminal|<---------+  PMSGLC  |<---------+Application|
| Screen |          |          |   Code   |  Program  |
+--------+          +----------+          +-----------+
```

Figure 3-8 Message Management Scenario for Error Handling

It is advisable to use the message management tools as much as possible since they allow you to change the contents of a message without changing application code. The use of PMSGLS should be limited to those occasions when the content of the message is determined at run-time.

The application message queue can be viewed using the <MESSAGE QUEUE> key (PF3) as described in the IISS Terminal Operator Guide. The Form Processor or System Message queue is displayed using the <SYSMSG> key in status mode.

3.8 Paging and Scrolling

Paging and scrolling is a capability of the FP that allows a form to contain more data than will fit on the screen at one time with provisions for viewing any portion of the data buffer. Paging and scrolling can be performed on those fields which have been defined as scrolling arrays. These are arrays which have been defined with an actual size greater than the display size. A scroll will advance/backup, by one element, the elements of the array that are displayed. A page is some multiple of scrolls, usually equal to the display size. An exception occurs when the displayed elements are within a display size of an actual size endpoint. In this case, a page will scroll to the end of the array. Scrollable arrays are especially useful for those applications which must display a list of objects from a database or file and the list is larger than the display area.

The paging and scrolling function keys are active when the FP is in "scrll/page" mode as described in the Terminal Operator Guide.

3.9 Logical Devices

A logical device is a conceptual device that is viewed by an application as a unique Display List. There can be more than one logical device on a physical device. An application is assigned one logical device when the routine INITFP (INITialize Form Processor) is called at start up. During processing, the application may create and manipulate other logical devices by calling the routines OPNLDV, CHGLDV, INQLDV, and CLSLDV. The logical device whose Display List is being manipulated is referred to as the current logical device.

The routine CHGLDV causes the logical device specified as an input parameter to become the current logical device. This logical device continues to be current until CHGLDV is called again and another device is named as an input parameter.

Logical devices provide the mechanism for implementing pop-up menus in an application. To do this, you would define application help for an item field. When the help key is pressed, the application could then open a new logical device, and display the appropriate menu of options for the field. When the user positions the cursor on the desired choice and presses the <ENTER> key, the application can determine the menu pick and continue processing. Logical devices are also used to distinguish between multiple applications running concurrently on a physical device.

3.10 Creating and Modifying Forms at Run-Time

Forms can be created and modified at run-time using FP routines. This provides a great deal of flexibility to the user interface of an interactive application. The routines that provide this capability are based on the syntax of the Form Definition Language. A thorough understanding of the material on defining forms presented in the Form Editor User's Manual is essential to using these routines.

To modify an existing form at run-time, the form must be open. The changes are made to the form on the Open List and then the changes are made to all instances of the form on the Display List.

SECTION 4

FP CALLABLE ROUTINES

This section of the manual describes the callable FP routines. They are listed alphabetically with a brief functional description.

ROUTINE	FUNCTION
ADDDIM	Add an array dimension to a field.
ADDELM	Add an element to the end of an open ended array.
ADDFLD	Add a field with specified characteristics to all instances of a form.
ADDFRM	Add a form to a window.
APRFLD	Make a field appear or disappear.
CHGLDV	Change the logical device associated with the current Display List.
CLSFRM	Close a form.
CLSLDV	Close a logical device.
CRTFLD	Add a field with default characteristics to all instances of a form.
CRTFRM	Create an empty form.
GDATA	Get data from a Display List element.
GDATLN	Get data length.
GDPFEX	Identify field whose value depends on the specified field.
GDPFLC	Identify field whose horizontal location depends on the specified field.
GETATT	Get an item attribute.
GETBAK	Get form or window background.
GETCUR	Get the current cursor position.
GETDQN	Get default qualified name.
GETLDV	Get logical device.
GETVTI	Get application data from the Virtual Terminal.
GFMFLD	Return the name and type of a field on a form.
GPAGE	Get form on page.
GTUINF	Get user information.
GTUSYM	Return prototype strings for IISSSLIB and IISSULIB.
GWINDO	Get window information.
INITFP	Initialize Form Processor.
INITVT	Set VTI mode flag.
INQABS	Return absolute row and column of a field location.
INQAPR	Return a field's appears if criterion.
INQATT	Return attribute or attribute primitive values defined for a form.
INQDIM	Return dimension information for a field array.
INQDIS	Return the current permanent attribute of a form or field.
INQDOM	Return an item field's domain values.
INQHLP	Return item field's help messages.
INQLDV	Inquire logical device.
INQLOC	Return relative location of a field.
INQPRO	Return field prompt information.
INQSIZ	Return field size information.

INQTYP	Return field type information.
INQVAL	Return field value expression information.
MAKFRM	Create a form with specified size and attribute.
MOVLDV	Move logical device
MVRFLD	Copy a field specifying name and location.
NUMELM	Return elements in open-ended array.
OISCR	Output a screen and wait for input.
OPNFRM	Open a form.
OPNLDV	Open a logical device.
OUTSCR	Output a screen.
PARFQN	Parse a qualified name.
PDATA	Put data in a Display List element.
PMSGLC	Put the message corresponding to the code into the message line.
PMSCLS	Put a string into the message line.
PUTATT	Put an item attribute.
PUTBAK	Put a form or window background.
PUTCUR	Put the cursor in a specified field position.
PUTLOC	Put the cursor in a specified location within a field.
PUTVTI	Send application data to Virtual Terminal.
REPFLD	Copy field with default location.
REPFRM	Copy form specifying new name.
RMVAPR	Remove a field's appears if criterion.
RMVARY	Remove all field array dimensions.
RMVATT	Remove form attribute.
RMVDIM	Remove a field array dimension.
RMVDOM	Remove item field domain options.
RMVFLD	Remove field from a form.
RMVHLP	Remove item field help.
RMVPAG	Remove a page from a window.
RMVPRO	Remove form or field prompt text.
RMVVAL	Remove an item field's value expression.
RPLFRM	Replace a form.
SAVFRM	Save dynamically created or modified forms.
SETAPR	Specify a field's appears if criterion.
SETATT	Set or change form attribute primitives.
SETDIM	Change a specified field array dimension.
SETDIS	Specify item field display attribute.
SETDOM	Specify item field domain options.
SETDQN	Set the default qualified name.
SETHLP	Specify item field help.
SETLDV	Specify logical device display parameters.
SETLOC	Specify field location.
SETNAM	Specify field name.
SETPRO	Add form or field prompt.
SETSIZ	Specify field size.
SETTYP	Specify field type.
SETVAL	Define an item field's value expression.
TERMFP	Terminate Form Processor.
TERMVT	Clear VTI mode flag.

The use of each routine is explained in detail in the following sections. This includes a description of the routine's purpose, the format for calling the routine, descriptions of the calling parameters, and an example call. Note that the calling

formats and examples are given in C, and Str and Num in the parameter format mean character string and long integer. Example data types in COBOL, PL1, C, and Fortran are:

	COBOL	PL1	C	FORTRAN
Num	S9(5) COMP	FIXED BIN(31)	int	INTEGER
Str	PIC X(n)	CHAR(N)	char[n]	CHARACTER*N

Note: A C character string may contain the null character if it is after the last character the FP looks at (i.e., after the semicolon in a qualified name).

4.1 ADDDIM

This routine is used to add an array dimension to a field at run-time. The routine parameters correspond to the components of the Repeat Specification syntax described in the Form Editor User's Manual.

4.1.1 Calling Format

```
CALL "ADDDIM" USING FORM-NAME,
                    FIELD-NAME,
                    DISPLAY-SIZE,
                    ACTUAL-SIZE,
                    DIRECTION,
                    SPACES,
                    RCODE.
```

4.1.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field to define an array dimension for.
DISPLAY-SIZE	Input	Num	The number of field repetitions to display on the form.
ACTUAL-SIZE	Input	Num	The total number of times the field actually repeats.
DIRECTION	Input	Str(1)	A character code that specifies the repeat direction. Values are H - Horizontal and V - Vertical.
SPACES	Input	Num	The number of spaces to leave between field repetitions.

RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.1.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME     PIC X(10) VALUE "TSTFRD".
01   DSP-SIZE        PIC S9(5) COMP VALUE 10.
01   ACT-SIZE        PIC S9(5) COMP VALUE 20.
01   NUM-SPACES      PIC S9(5) COMP VALUE 0.
01   RPT-DIRECTION   PIC X VALUE "V".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "ADDDIM" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         DSP-SIZE,
                         ACT-SIZE,
                         NUM-SPACES,
                         RPT-DIRECTION,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.2 ADDELM

This routine is used to add an element to the end of an open-ended array. An open-ended array is one which is defined with length * instead of an integer number of rows. The Form Editor User's Manual describes the syntax for defining arrays.

4.2.1 Calling Format

```
CALL "ADDELM" USING QUALIFIED-NAME,
                    ELEMENT-NUMBER,
                    RCODE.
```

4.2.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-NAME	Input	Str(1-120)	The name of the array that you want to add the element to.
ELEMENT-NUMBER	Output	Num	The number of the element that was added to the array.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.2.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   PATH-NAME      PIC X(11) VALUE "EXPFRM.ARY;".
01   ELEMENT-NUMBER PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "ADDELM" USING PATH-NAME,
                         ELEMENT-NUMBER,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.3 ADDFLD

This routine is used to add a field to a form at run-time with the location, size, type, and display attribute you specify. The field is added to all instances of the form currently on the Display List. It will also appear on any subsequent instances of the form added to the Display List.

4.3.1 Calling Format

```
CALL "ADDFLD" USING FORM-NAME,
                    FIELD-NAME,
                    FIELD-TYPE,
                    ATTRIBUTE-ID,
                    WIDTH,
                    DEPTH,
                    VREL-FIELD,
                    VEXTREF,
                    VINTREF,
                    ROW,
                    HREL-FIELD,
                    HEXTREF,
                    HINTREF,
                    COLUMN,
                    RCODE.
```

4.3.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form that the new field is being added to.
FIELD-NAME	Input	Str(10)	The name of the field being added.
FIELD-TYPE	Input	Str(1)	A character code to identify the type of field being added. Values are: W - window, F - form, and I - item.
ATTRIBUTE-ID	Input	Str(10)	The attribute value to be associated with the new field. This can be a user-defined attribute or one of the predefined values defined in the include member FPPARM.
WIDTH	Input	Num	The number of columns wide the field is.
DEPTH	Input	Num	The number of rows high the field is.
VREL-FIELD	Input	Str (1-120)	The qualified name of the field that FIELD-NAME's position is vertically relative to or blank if it is relative to its containing form.
VEXTREF	Input	Num	The vertical reference point on VREL-FIELD that FIELD-NAME's position is relative to. Values are 1 - top, 2 - center, and 3 - bottom.
VINTREF	Input	Num	The vertical reference point of FIELD-NAME that its position is relative to. Values are the same as for VEXTREF.

ROW	Input	Num	The number of rows that FIELD-NAME's position is offset from VREL-FIELD's reference point.
HREL-FIELD	Input	Str (1-120)	The qualified name of the field that FIELD-NAME's position is horizontally relative to or blank if it is relative to the containing form.
HEXTREF	Input	Num	The horizontal reference point on HREL-FIELD that FIELD-NAME's position is relative to. Values are 1 - left, 2 - center, and 3 - right.
HINTREF	Input	Num	The horizontal reference point of FIELD-NAME that the position is relative to. Values are the same as for HEXTREF.
COLUMN	Input	Num	The number of columns the field is offset from HREL-FIELD's reference point.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.3.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC  X(10)        VALUE "EXPFRM".
01   NEWFLD-NAME     PIC  X(10)        VALUE "NEWITEM".
01   NEWFLD-TYPE     PIC  X(1)         VALUE "I".
01   NEWFLD-WIDTH    PIC S9(5)   COMP  VALUE 10.
01   NEWFLD-DEPTH    PIC S9(5)   COMP  VALUE 1.
01   REF-FLD-NAME    PIC  X(7)         VALUE "TSTFLD;".
01   TOP-LEFT-REF    PIC S9(5)   COMP  VALUE 1.
01   CENTER-REF      PIC S9(5)   COMP  VALUE 2.
01   BOT-RIGHT-REF   PIC S9(5)   COMP  VALUE 3.
01   ZERO-COMP       PIC S9(5)   COMP  VALUE 0.
01   TWO-COMP        PIC S9(5)   COMP  VALUE 2.
```

```
PROCEDURE DIVISION.
     .
     .
     .
     CALL "ADDFLD" USING EXPFRM-NAME,
                         NEWFLD-NAME,
                         NEWFLD-TYPE,
                         INP,
                         NEWFLD-WIDTH,
                         NEWFLD-DEPTH,
                         REF-FLD-NAME,
                         BOT-RIGHT-REF,
                         TOP-RIGHT-REF,
                         TWO-COMP,
                         REF-FLD-NAME,
                         CENTER-REF,
                         CENTER-REF,
                         ZERO-COMP,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.4 ADDFRM

This routine adds a form to a window. The window you are adding the form to must already be on the Display List. If the form is not already open, it will be opened automatically. Adding a form to a window creates a new page.

4.4.1 Calling Format

```
CALL "ADDFRM" USING QUALIFIED-WINDOW-NAME,
                    FORM-NAME,
                    PAGE-NUMBER,
                    RCODE.
```

4.4.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-WINDOW-NAME	Input	Str(1-120)	The qualified name of the Display List element that identifies the window that will contain the form.
FORM-NAME	Input	Str (10)	The name of the form being added.
PAGE-NUMBER	Output	Num	The number of the page that the added form creates in the window.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.4.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   PAGE-NUMBER    PIC S9(5) COMP.
01   EXPFRM-NAME    PIC X(10) VALUE "EXPFRM".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "ADDFRM" USING SCREN,
                         EXPFRM-NAME,
                         PAGE-NUMBER,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.5 APRFLD

This routine allows you to control when a field appears or does not appear on its containing form. The APR-FLAG can be set to display the field, not display the field, or toggle (i.e., do the opposite of what the current setting is). The effect of this routine takes place at the next OISCR. If a loop that contains a call to OISCR is preceded by a call to APRFLD to toggle the setting, the setting will remain the same for all the OISCR calls. If the call to APRFLD is also inside the loop, the APR-FLAG setting will toggle for each call to OISCR. NOTE that any appears if criterion defined for the field is evaluated during an OISCR and therefore, will cancel the APR-FLAG setting. If an item field has a location of 0 0 (i.e., it is non-display), a location must be defined for the field before APRFLD can be called to display the field.

4.5.1 Calling Format

```
CALL "APRFLD" USING QUALIFIED-NAME,
                    APR-FLAG,
                    RCODE.
```

4.5.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-NAME	Input	Str(1-120)	The qualified name of the Display List element that identifies the field.
APR-FLAG	Input	Str(1)	A character code to specify the appear flag setting. Values are A - Appears, D - Disappear, and T - Toggle.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.5.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   FLAG            PIC X.
01   PATH-NAME       PIC X(7) VALUE "MYFORM;".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "APRFLD" USING PATH-NAME,
                         FLAG,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.6 CHGLDV

This routine is used to make the Display List associated with the specified logical device the current Display List. After CHGLDV is called, all form processing is done on the named logical device until it is called again with a different logical-device-id.

NOTE: If you are changing from the initial logical device assigned at start up and you intend to return processing to it at a later time, you need to call INQLDV to determine its logical-device-id before you call CHGLDV.

4.6.1 Calling Format

```
CALL "CHGLDV" USING LOGICAL-DEVICE-ID,
                    RCODE.
```

4.6.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
LOGICAL-DEVICE-ID	Input	Num	The ID of the logical device that you want to be the current display list. The logical-device-id is returned when you call OPNLDV to create a logical device.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.6.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   LOGICAL-DEVICE-ID   PIC S9(5)  COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "CHGLDV" USING
LOGICAL-DEVICE-ID,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.7 CLSFRM

This routine is used to close a form. This means that the memory that was used by this form while the application had it open (see OPNFRM) is released so that it may be allocated to another form. In order to close a form, it must not be in a window (i.e., it was never in a window, it was removed from the window using RMVPAG, or it was replaced using RPLFRM) and it must not be a subform on any other open forms. NOTE that if an open form is changed at run-time, the routine SAVFRM must be called before closing it to save the changes.

4.7.1 Calling Format

```
CALL "CLSFRM" USING FORM-NAME,
                    RCODE.
```

4.7.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form being closed.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.7.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME    PIC X(10) VALUE "EXPFRM".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "CLSFRM" USING EXPFRM-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
     CALL "PMSGLC" USING RCODE.
```

4.8 CLSLDV

CLSLDV is used to close a logical device. When a logical device is closed, the Display List associated with it is removed from memory. You cannot close the original logical device that is created by INITFP or the logical device associated with the current Display List. In that case, you must call CHGLDV to associate a different device with the current Display List.

4.8.1 CALLING FORMAT

```
CALL "CLSLDV" USING LOGICAL-DEVICE-ID,
                    RCODE.
```

4.8.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
LOGICAL-DEVICE-ID	Input	Str(10)	The id of the logical device that you want to close. You obtain the logical-device-id of the device when you create the device using OPNLDV.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.8.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   LOGICAL-DEVICE-ID   PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "CLSLDV" USING
LOGICAL-DEVICE-ID,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.9 CRTFLD

This routine adds a field to all instances of an open form at run-time. Required field characteristics have default values. The field is a non-displayed ITEM. This means that the location must be specified using the routine SETLOC in order for the item to appear on the screen. The size of the field defaults to one and its display attribute is text. Other routines may be used to change these defaults or further define the field to include such things as prompts, help, and array dimensions.

4.9.1 Calling Format

```
CALL "CRTFLD" USING     FORM-NAME,
                          FIELD-NAME,
                          RCODE.
```

4.9.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form that the new field is being added to.
FIELD-NAME	Input	Str(10)	The name of the field that is being added to the form.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.9.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME    PIC X(10) VALUE "EXPFRM".
01   NEWFLD-NAME    PIC X(10) VALUE "NEWFLD".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "CRTFLD" USING EXPFRM-NAME,
                         NEWFLD-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.10 CRTFRM

This routine creates a new form at run-time. The background attribute is transparent and the size defaults to 0 0 meaning it is not a fixed size. Other routines may be used to override these defaults or further define the form to include such things as prompts, fields, and particular attributes. The form is open.

4.10.1 Calling Format

```
CALL "CRTFRM" USING FORM-NAME,
                    RCODE.
```

4.10.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the new form being created.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.10.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   NEWFRM-NAME    PIC X(10) VALUE "NEWFORM".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "CRTFRM" USING NEWFRM-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.11 GDATA

This routine is used to access user entered data from an element on the current Display List. The location of the data is specified by a qualified name as explained in section 3.4 of this manual. The specified element cannot be a window. This means that the data must be from a form or an item. If the Display List element is an array, you will get the data from all the array elements unless you specify a specific one. You can also specify whether you want the current or previous instance of the data. By comparing the current and previous instance of data, you can find out what the user has changed on a form.

4.11.1 Calling Format

```
CALL "GDATA" USING    INSTANCE-ID,
                      QUALIFIED-NAME,
                      OUTPUT-DATA,
                      RCODE.
```

4.11.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
INSTANCE-ID	Input	Str(10)	The instance of the data you want to access. The include file FPPARM contains the definition for the instance-id values PREV and CURRNT.
QUALIFIED-NAME	Input	Str (1-120)	The qualified name that identifies the Display List element whose data you want to access. The routines GWINDO and GPAGE may have to be used to help you define the name.
OUTPUT-DATA	Output	Var.	The data structure containing the accessed data. You have the option of defining the structure in your application or using the structure defined forms definition tool MAKINC described in the Form Editor UM

RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.11.3 Example

```
    WORKING-STORAGE SECTION.
    COPY FPCODE OF IISSCLIB.
    COPY FPPARM OF IISSCLIB.
    COPY EXPFRM OF IISSCLIB.
  *EXPLANATION OF COPY MEMBER EXPFRM
  *01    EXPFRM.
  * 02 FIELD1            PIC X(6).
  * 02 FIELD2            PIC X(10).
  * 02 FIELD3            PIC X(2).
    01   EXPFRM-NAME     PIC X(7) VALUE "EXPFRM;".
    PROCEDURE DIVISION.
         .
         .
         .
         CALL "GDATA" USING  CURRNT,
                             EXPFRM-NAME,
                             EXPFRM,
                             RCODE.
         IF RCODE IS EQUAL TO OK
         .
         .
         .
         ELSE
            CALL "PMSGLC" USING RCODE.
```

4.12 GDATLN

This routine is used to determine the length of data contained in a Display List. For instance, GDATLN can be called to get the length of the data that would be returned by a call to GDATA or that would be written by a call to PDATA.

4.12.1 Calling Format

```
CALL "GDATLN" USING QUALIFIED-NAME,
                    DATA-LENGTH,
                    RCODE.
```

4.12.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
QUALIFIED-MAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the data you want the length of.
DATA-LENGTH	Output	Num	The length of the specified data.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.12.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   DATA-LEN        PIC S9(5) COMP.
01   PATH-NAME       PIC X(7) VALUE "MYFORM;".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GDATLN" USING PATH-NAME,
                         DATA-LEN,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.13 GDPFEX

This routine is used to find the names of fields on a specified form whose values depend on a particular field on the form.

4.13.1 Calling Format

```
CALL "GDPFEX" USING FORM-NAME,
                    FIELD-NAME,
                    FIELD-NUM,
                    DEP-FIELD,
                    RCODE.
```

4.13.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
FORM-NAME	Input	Str(10)	The name of the form that contains the field that other fields' values depend on.
FIELD-MAME	Input	Str(10)	The name of the field that other field's values depend on.
FIELD-NUM	Input	Num	A number which indicates which field on the form.
DEP-FIELD	Output	Str(10)	The name of the field whose value depends on the indicated field.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.13.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME     PIC X(10) VALUE "TSTFLD".
01   FIELD-NUM       PIC S9(5) COMP.
01   DEPFLD-NAME     PIC X(10).
PROCEDURE DIVISION.
     .
     .
     .
     PERFORM GET-NEXT VARYING FIELD-NUM FROM ZERO BY 1
                      UNTIL RCODE NOT EQUAL TO OK.
     .
     .
     .
GET-NEXT.
     CALL "GDPFEX" USING EXPFRM-NAME,
                      TSTFLD-NAME,
                      FIELD-NUM,
                      DEPFLD-NAME,
                      RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.14 GDPFLC

This routine is used to find the names of fields on a specified form whose locations depend on a particular field on the form.

4.14.1 Calling Format

```
CALL "GDPFLC" USING FORM-NAME,
                    FIELD-NAME,
                    FIELD-NUM,
                    DEP-FIELD,
                    RCODE.
```

4.14.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
FORM-NAME	Input	Str(10)	The name of the form that contains the field that other fields' values depend on.
FIELD-MAME	Input	Str(10)	The name of the field that other field's values depend on.
FIELD-NUM	Input	Num	A number which indicates which field on the form.
DEP-FIELD	Output	Str(10)	The name of the field whose value depends on the indicated field.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.14.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME     PIC X(10) VALUE "TSTFLD".
01   FIELD-NUM       PIC S9(5) COMP.
01   DEPFLD-NAME     PIC X(10).
PROCEDURE DIVISION.
     .
     .
     .
     PERFORM GET-NEXT VARYING FIELD-NUM FROM ZERO BY 1
                      UNTIL RCODE NOT EQUAL TO OK.
     .
     .
     .
GET-NEXT.
     CALL "GDPFLC" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         FIELD-NUM,
                         DEPFLD-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.15 GETATT

The foreground attribute for items (specified during form definition with the DISPLAY AS clause), can be changed at run-time using the routine PUTATT. Foreground attributes can be changed for the next display only (temporary) or until PUTATT is called again (permanent). This routine allows you to find out the current values for temporary and permanent attributes.

4.15.1 Calling Format

```
CALL "GETATT" USING QUALIFIED-NAME,
                    ATTRIBUTE-TYPE,
                    ATTRIBUTE-ID,
                    RCODE.
```

4.15.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
QUALIFIED-NAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the item whose attribute you want to know. Refer to Section 3.
ATTRIBUTE-TYPE	Input	Num	The attribute type whose value you want to know. The include FPPARM contains the definitions for the type values PERM & TEMP
ATTRIBUTE-ID	Output	Str(10)	The attribute value defined for the specified item for the specified attribute type. If a temporary attribute has not been defined or has already been displayed, this value will be blanks.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.15.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   ITEM-PATH       PIC X(12) VALUE "FORM1.ITEM2;".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GETATT" USING ITEM-PATH,
                         PERM,
                         ATTRIB-ID,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.16 GETBAK

This routine allows you to find out the current values for form and window background attributes. The background attribute of forms and windows (specified during form definition with the BACKGROUND clause or with the routine SETATT), can be changed at run-time using the routine PUTBAK. Some attributes can be changed for the next display only (temporary) or until the routine PUTBAK is called again (permanent). However, temporary is not supported at this time and an attribute value remains in effect for all following calls to OUTSCR or OISCR until PUTBAK is called again.

4.16.1 Calling Format

```
CALL "GETBAK" USING QUALIFIED-NAME,
                    BACKGROUND-TYPE,
                    BACKGROUND-ID,
                    RCODE.
```

4.16.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
QUALIFIED-NAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the form or window whose background you want to know. Refer to Section 3.4.
BACKGROUND-TYPE	Input	Num	The background type whose value you want to know. The include member contains the definition for type values PERM & TEMP. TEMP is not supported at this time.
BACKGROUND-ID	Output	Str(10)	The background value defined for the specified form or window.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.16.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   BACK-ID        PIC X(10).
01   QUAL-NAME      PIC X(11) VALUE "FORM1.WIN2;".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GETBAK" USING QUAL-NAME,
                         PERM,
                         BACK-ID,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.17 GETCUR

This routine finds the current cursor position. It returns the qualified name of the Display List element in which the cursor is found and the row and column within this element. The subscripts (if any) of an array in which it lies as well as page numbers are included in the fully qualified name. In applications that have menu selection by cursor position, this routine is used to determine the user's menu choice.

4.17.1 Calling Format

```
CALL "GETCUR" USING QUALIFIED-NAME,
                    FIELD-TYPE,
                    ROW,
                    COL,
                    RCODE.
```

4.17.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
QUALIFIED-NAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the current position of the cursor. See Section 3.4.
FIELD-TYPE	Input	Num	A character code to identify the type of field the cursor was found in. Values are: F - form, W - window, and I - item.
ROW	Output	Num	The row position of the cursor within the field (i.e. with respect to the origin of the particular field).
COL	Output	Num	The column position of the cursor within the field (i.e. with respect to the origin of the particular field).

RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.17.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   FULL-NAME       PIC  X(120).
01   TYPE1           PIC  X.
01   ROW             PIC S9(5) COMP.
01   COL             PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GETCUR" USING FULL-NAME,
                         TYPE1,
                         ROW,
                         COL,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.18 GETDQN

This routine gets the fully qualified name which identifies the current default field. A procedure which handled a form might get and save the current default qualified name, set the default to its own form and restore the previous default with the saved name before returning.

4.18.1 Calling Format

```
CALL "GETDQN" USING FULL-NAME,
                    RCODE.
```

4.18.2 Parameter Descriptions

FULL-NAME	Output	Str (120)	The absolute fully qualified name that identifies the default field for relative qualified names.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.18.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   FULL-NAME      PIC X(120).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GETDQN" USING FULL-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.19 GETLDV

This routine gets the display parameters (row, col, width and depth) for a logical device belonging to an application. If the logical device is not opened GETLDV will return an NFPDSTRC error.

4.19.1 Calling Format

```
CALL "GETLDV" USING LDWNID,
                    DSPROW,
                    DSPCOL,
                    DSPWDTH,
                    DSPDPTH,
                    RCODE.
```

4.19.2 Parameter Description

NAME	I/O	FORMAT	DESCRIPTION
LDWNID	Input	Num	The id of the logical device.
DSPROW	Output	Num	The row position of the origin of the logical device's display
DSPCOL	Output	Num	The column position of the origin of the logical device's display.
DSPWDTH	Output	Num	The width of the logical devices's display.
DSPDPTH	Output	Num	The depth of the logical device's display.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.19.3 Example

```
WORKING_STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GETLDV" USING LDWNID,
                         DSPROW,
                         DSPCOL,
                         DSPWDTH,
                         DSPDPTH,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.20 GETVTI

This routine gets application data from the Virtual Terminal. The routine INITVT must be called first to enable VT pass-through mode.

4.20.1 Calling Format

```
CALL "GETVTI" USING BUFFER,
                    BUF-SIZE,
                    LENGTH,
                    CODE.
```

4.20.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
BUFFER	Output	Str(1-BUF-SIZE)	The data and commands received from the Virtual Terminal.
BUF-SIZE	Input	Num	The size of BUFFER.
LENGTH	Output	Num	The actual length of the data and commands returned in BUFFER
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.20.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   VTI-BUF        PIC X(1) OCCURS 1 TO 1000 TIMES
                    DEPENDING ON BUF-LEN.
01   BUF-SIZ        PIC S9(5) COMP VALUE 1000.
01   BUF-LEN        PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INITVT" USING RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
        CALL "GETVTI" USING VTI-BUF,
                            BUF-SIZ,
                            BUF-LEN,
                            RCODE
        IF RCODE IS EQUAL TO OK
     .
     .
     .
        ELSE
            CALL "PMSGLC" USING RCODE
     ELSE
         CALL "PMSGLC" USING RCODE.
```

4.21 GFMFLD

This routine returns the name and type of a field on a specified form based on the order fields were defined on the form. To return this information for all fields on a form, this routine should be called in a loop that is terminated by the warning NOMORFLD. In order to determine complete dependencies, if the form contains form fields, their field information should also be obtained.

4.21.1 Calling Format

```
CALL "GFMFLD" USING FORM-NAME,
                    FIELD-NUM,
                    FIELD-NAME,
                    FIELD-TYPE,
                    RCODE.
```

4.21.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
FORM-NAME	Input	Str(10)	The name of the form whose field names you want.
FIELD-NUM	Input	Num	A number based on the order fields were defined on the form to indicate the field whose name you want (i.e. a positive number of 1 or more refers to the actual order the fields were defined in. 0 refers to the most recently defined field and a negative number of 1 or more refers to the reverse order the fields were defined starting with the next most recently added.
FIELD-NAME	Output	Str(10)	The name of the field referred to by the specified FIELD-NAME.

FIELD-TYPE	Output	Str(1)	A character code to identify the type of field FIELD-NAME is. Values are: F - form, W - window, and I - item.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.21.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC  X(10) VALUE "EXPFRM".
01   FIELD-NUM       PIC S9(5)  COMP.
01   FIELD-NAME      PIC  X(10).
01   FIELD-TYPE      PIC  X(1).
PROCEDURE DIVISION.
     .
     .
     .
     PERFORM GET-NEXT VARYING FIELD-NUM FROM ZERO BY 1
                      UNTIL RCODE NOT EQUAL TO OK.
     .
     .
     .
GET-NEXT.
     CALL "GFMFLD" USING EXPFRM-NAME,
                      FIELD-NUM,
                      FIELD-NAME,
                      FIELD-TYPE,
                      RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.22 GPAGE

This routine finds the name of the form that a specified page of a window currently contains. The routine GWINDO may be called to find out how many pages are currently contained in a window. This information is used to determine qualified names in the current Display List.

4.22.1 Calling Format

```
CALL "GPAGE" USING  QUALIFIED-WINDOW-NAME,
                    PAGE-NUMBER,
                    FORM-NAME,
                    RCODE.
```

4.22.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
QUALIFIED-WINDOW-NAME	Input	Str (1-120)	The qualified name of the window whose contents you are identifying.
PAGE-NUMBER	Input	Num	The page of the window that contains the form you are identifying.
FORM-NAME	Output	Str(10)	The name of the form that the specified page of the window contains.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.22.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC X(10).
01   CURPAG          PIC S9(5) COMP VALUE 2.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GPAGE" USING  SCREN,
                         CURPAG,
                         EXPFRM-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
ELSE
   CALL "PMSGLC" USING RCODE.
```

4.23 GTUINF

This routine returns the values that were in the User ID and Role fields on the IISS Function Screen when the application began. This information may be used to control the application. For example to access a shared data base you could select the data in the data base using the user's name. The user's role can be used to allow different users access to a selected group of functions.

4.23.1 Calling Format

```
CALL "GTUINF" USING USRNAM,
                    USROLE,
                    RCODE.
```

4.23.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
USRNAM	Output	Str(10)	User's Name
USROLE	Output	Str(10)	User's Role
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.23.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   USRNAM          PIC X(10)
01   USROLE               PIC X(10)
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GTUINF" USING                          USRNAM,
                         USROLE,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.24 GTUSYM

This routine returns the prototype strings for the location of the source and compiled forms (IISSSLIB and IISSULIB) as defined in the UI Database for the logged on user.

4.24.1 Calling Format

```
CALL "GTUSYM" USING SLIB,
                    ULIB,
                    RCODE.
```

4.24.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
SLIB	Output	Str(80)	The prototype string for the form source file.
ULIB	Output	Str(80)	The prototype string for the compiled form file.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.24.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   SLIB                PIC X(80)
01   ULIB           PIC X(80)
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GTUSYM" USING SLIB,
                         ULIB,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.25 GWINDO

This routine lets you find how many pages a window currently contains. The routine GPAGE can then be called to find the name of the form contained in a page. This information is used to determine qualified names of elements in the current Display List.

4.25.1 Calling Format

```
CALL "GWINDO" USING QUALIFIED-WINDOW-NAME,
                    TOTAL-PAGES,
                    RCODE.
```

4.25.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
QUALIFIED-WINDOW-NAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the window whose page count you want.
TOTAL-PAGES	Output	Num	The total number of pages the specified window currently contains.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.25.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   TOTAL-PAGES    PIC S9(5) COMP.
PROCEDURE DIVISION.
     CALL "GWINDO" USING SCREN,
                         TOTAL-PAGES,
     RCODE.
IF RCODE IS EQUAL TO OK
     .
     .
     .
ELSE
   CALL "PMSGLC" USING RCODE.
```

4.26 INITFP

This routine is used to initialize the Form Processor. This includes creating a logical device which consists of a Display List beginning with the form PSCREN. INITFP must be called prior to any other FP routines.

4.26.1 Calling Format

CALL "INITFP".

4.26.2 Parameter Descriptions

NONE.

4.26.3 Example

```
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INITFP".
```

4.27 INITVT

This routine sets the VT pass-through mode which provides direct access to the Virtual Terminal without using the Form Processor. The routines GETVTI and PUTVTI are then used to receive and transmit data. This mode is primarily for system code and should not be used in user applications.

4.27.1 Calling Format

```
CALL "INITVT" USING RCODE.
```

4.27.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.27.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INITVT" USING RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.28 INQABS

This routine returns the absolute row and column of a field's position relative to its containing form after resolving relative positions.

4.28.1 Calling Format

```
CALL "INQABS" USING FORM-NAME,
                    FIELD-NAME,
                    ROW,
                    COL,
                    RCODE.
```

4.28.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose absolute location you want.
ROW	Output	Num	The absolute row of the field location.
COL	Output	Num	The absolute column of the field location.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.28.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME     PIC X(10) VALUE "TSTFLD".
01   TSTFLD-ROW      PIC S9(5) COMP.
01   TSTFLD-COL      PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQABS" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         TSTFLD-ROW,
                         TSTFLD-COL,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.29 INQAPR

This routine returns the criterion defined to determine when the specified field appears on its containing form. The CRITERION parameter corresponds to the Criterion component of the APPEARS IF clause syntax described in the Form Editor User's Manual.

4.29.1 Calling Format

```
CALL "INQAPR" USING FORM-NAME,
                    FIELD-NAME,
                    BUF-SIZE,
                    CRITERION,
                    LENGTH,
                    RCODE.
```

4.29.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose appears if criterion you want.
BUF-SIZE	Input	Num	The length of the buffer being reserved for the criterion.
CRITERION	Output	Str (BUF-SIZE)	The appears if criterion defined for the field either padded with spaces or truncated
LENGTH	Output	Num	The actual length of the criterion without padding. Also the minimum buffer length needed to receive a non-truncated criterion. Comparing BUF-SIZE and LENGTH determines if CRITERION is truncated.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

This routine may first be called with a BUF-SIZE of zero. This forces an error but LENGTH will be the actual buffer size needed to store the appears if criterion.

4.29.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME     PIC X(10) VALUE "TSTFLD".
01   BUF-SIZE        PIC S9(5) COMP VALUE 100.
01   CRIT-BUF        PIC X(100).
01   BUF-LEN         PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQAPR" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         BUF-SIZE,
                         CRIT-BUF,
                         BUF-LEN,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.30 INQATT

This routine returns either an attribute name or the primitive values for a specified form attribute. The routine parameters correspond to the components of the ATTRIBUTE clause syntax described in the Form Editor User's Manual.

4.30.1 Calling Format

```
CALL "INQATT" USING FORM-NAME,
                    ATTRIBUTE-ID,
                    ATTR-TYPE,
                    INDEX,
                    ATTRIBUTE,
                    RCODE.
```

4.30.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
FORM-NAME	Input	Str(10)	The name of the form whose attribute information you want.
ATTRIBUTE-ID	Input	Str(10)	The name of the attribute whose primitives you want or blank to indicate that you want the name of the attribute specified by the INDEX value.
ATTR-TYPE	Input	Str(1)	A character code to identify which type of primitive you want. Values are: D - display color, B - background color, & P - other primitives, such as hidden, guarded, etc. ATTR-TYPE shuld be blank if ATTRIBUTE-ID is blank.

INDEX	Input	Num	An index number to specify position in a list. If ATTRIBUTE-ID is blank, this is the list of all attributes defined for the form. If ATTRIBUTE-ID is not blank, this is the list of primitives for the specified attribute. INDEX is ignored the ATTR-TYPE value is "B" or "D".
ATTRIBUTE	Output	Str(10)	The name of the attribute if ATTRIBUTE-ID is blank or the attribute primitive (or color if ATTRIBUTE-ID is not blank). The string is non-null terminated and padded with blanks as needed.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.28.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01   TSTATR-NAME     PIC X(10) VALUE "TESTATTR".
01   DISP-TYPE       PIC X VALUE "D".
01   DUMMY           PIC S9(5) COMP.
01   DISP-COLOR      PIC X(10).
PROCEDURE DIVISION.
     .
     .
     .
```

INQATT

```
CALL "INQATT" USING EXPFRM-NAME,
                    TSTATR-NAME,
                    DISP-TYPE,
                    DUMMY,
                    DISP-COLOR,
                    RCODE.
IF RCODE IS EQUAL TO OK
.
.
.
ELSE
   CALL "PMSGLC" USING RCODE.
```

4.31 INQDIM

This routine returns information about a field's array dimensions.

4.31.1 Calling Format

```
CALL "INQDIM" USING FORM-NAME,
                    FIELD-NAME,
                    INDEX,
                    DISPLAY-SIZ,
                    ACTUAL-SIZ,
                    DIRECTION,
                    SPACES,
                    RCODE.
```

4.31.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose array dimension information you want.
INDEX	Input	Num	A number to indicate which array dimension information you want.
DISPLAY-SIZE	Output	Num	The number of field repetitions to display on the form. Zero is returned for open-ended arrays.
ACTUAL-SIZE	Output	Num	The total number of times the field actually repeats. Zero is returned for open-ended arrays.
DIRECTION	Output	Str(1)	The direction (H or V) of the the repeat.

SPACES	Output	Num	The number of spaces between elements.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.31.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME     PIC X(10) VALUE "TSTFLD".
01   TSTFLD-DIM      PIC S9(5) COMP VALUE 1.
01   DSP-SIZE        PIC S9(5) COMP.
01   ACT-SIZE        PIC S9(5) COMP.
01   NUM-SPACES      PIC S9(5) COMP.
01   RPT-DIRECTION   PIC X.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQDIM" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         TSTFLD-DIM,
                         DSP-SIZE,
                         ACT-SIZE,
                         NUM-SPACES,
                         RPT-DIRECTION,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.32 INQDIS

This routine returns the permanent attribute of a form or field. The permanent attribute corresponds to the attribute_id component in the BACKGROUND clause syntax for forms and window fields and in the DISPLAY AS clause syntax for item fields as described in the Form Editor User's Manual.

4.32.1 Calling Format

```
CALL "INQDIS" USING FORM-NAME,
                    FIELD-NAME,
                    PERM-ATTR,
                    RCODE.
```

4.32.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
FORM-NAME	Input	Str(10)	The name of the form whose attribute you want or that contains the field whose attribute you want.
FIELD-NAME	Input	Str(10)	The name of the field whose attribute you want or blank if you want the attribute of FORM-NAME.
PERM-ATTR	Output	Str(10)	The name of the form or field permanent attribute.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.32.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE FROM IISSCLIB.
COPY FPPARM FROM IISSCLIB.
01   EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME     PIC X(10) VALUE "TSTFLD".
01   ATTR-NAME       PIC X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQDIS" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         ATTR-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.33 INQDOM

This routine returns an item field's domain values. These values correspond to the options of the DOMAIN clause syntax described in the Form Editor User's Manual. The user can request a specific domain value or all the domain values defined for the item with the values separated by commas.

4.33.1 Calling Format

```
CALL "INQDOM" USING FORM-NAME,
                    ITEM-NAME,
                    INDEX,
                    BUF-SIZE,
                    DOMAIN,
                    LENGTH,
                    RCODE.
```

4.33.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
FORM-NAME	Input	Str(10)	The name of the containing form
ITEM-NAME	Input	Str(10)	The name of the item field whose domain values you want.
INDEX	Input	Num	A number to indicate which domain value you want. An INDEX follows the parameter descriptions.
BUF-SIZE	Input	Num	The size of the buffer being reserved for the domain value.
DOMAIN	Output	Str (BUF-SIZE)	The domain value or values, either padded with spaces or truncated.

LENGTH	Output	Num	The actual length of the DOMAIN clause without padding. This is also the minimum buffer length needed to receive a non-truncated domain clause. Comparing LENGTH with BUF-SIZE determines whether DOMAIN contains a truncated or padded value.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

INDEX Key

1 LEFT,RIGHT, MUST FILL or error
2 UPPER, LOWER or error
3 NUMERIC or error
4 MUST ENTER or error
5 MINIMUM int or error
6 MAXIMUM int or error
7 all separated by commas

For INDEX 7, this routine may first be called with a BUF-SIZE of zero. This causes an error but the value returned in LENGTH will be the actual buffer size needed to store all the domain values.

4.33.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01    EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01    TSTFLD-NAME     PIC X(10) VALUE "TSTFLD".
01    ALLDOM-INDEX    PIC S9(5) COMP VALUE 7.
01    BUF-SIZE        PIC S9(5) COMP VALUE 200.
01    DOMAIN-BUF      PIC X OCCURS 1 TO 200 TIMES
                      DEPENDING ON DOMAIN-LEN.
01    DOMAIN-LEN      PIC S9(5) COMP.
```

```
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQDOM" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         ALLDOM-INDEX,
                         BUF-SIZE,
                         DOMAIN-BUF,
                         DOMAIN-LEN,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.34 INQHLP

This routine returns the help information defined for an item field. The output parameters correspond to the components of the HELP clause syntax described in the Form Editor User's Manual.

4.34.1 Calling Format

```
CALL "INQHLP" USING FORM-NAME,
                    ITEM-NAME,
                    TYPE,
                    HELP,
                    RCODE.
```

4.34.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
FORM-NAME	Input	Str(10)	The name of the containing form
ITEM-NAME	Input	Str(10)	The name of the item field whose help information you want
TYPE	Input	Str(1)	A character code to identify the type of help that is defined. Values are S- string, A - application, and F - form.
HELP	Output	Str(60)	The help information defined for the item field. Contents depend on the TYPE field. For F HELP is the name of the help form, for A it is the word "APPLICATION", and for S it is the actual help text.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.34.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME     PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME     PIC X(10) VALUE "TSTFLD".
01   HELP-TYPE       PIC X.
01   HELP-MSG        PIC X(60).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQHLP" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         HELP-TYPE,
                         HELP-MSG,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.35 INQLDV

This routine is used to retrieve the logical device associated with the current Display List. NOTE: The id of the logical device that is assigned to an application when INITFP is called can only be retrieved by calling this routine. If the application uses multiple logical devices, it should call INQLDV before changing logical devices so that the original logical device can be used again.

4.35.1 Calling Format

```
CALL "INQLDV" USING      LOGICAL-DEVICE-ID,
                 RCODE.
```

4.35.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
LOGICAL-DEVICE-ID	Output	Num	The id of the logical device that the application is using for its current Display List.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.35.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   LOGICAL-DEVICE-ID   PIC S9(5)  COMP.
PROCEDURE DIVISION.
     .
     .
     .
CALL "INQLDV" USING      LOGICAL-DEVICE-ID,
                 RCODE.
IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.36 INQLOC

This routine returns the relative position information of a form or field. The output parameters correspond to the components of the Location syntax described in the Form Editor User's Manual.

4.36.1 Calling Format

```
CALL "INQLOC" USING      FORM-NAME,
                         FIELD-NAME,
                         VREL-FIELD,
                         VEXTREF,
                         VINTREF,
                         ROW,
                         HREL-FIELD,
                         HEXTREF,
                         HINTREF,
                         COLUMN,
                         RCODE.
```

4.36.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form or the name of the form whose position information you want.
FIELD-NAME	Input	Str(10)	The name of the field whose position information you want or blank if you want the position information for FORM-NAME.

VREL-FIELD	Output	Str (120)	The qualified name of the field that the form or field position is vertically relative to or blank if it is relative to its containing form.
VEXTREF	Output	Num	The vertical reference point on VREL-FIELD that the form or field location is relative to. Values are 1 - top, 2 - center, 3 - bottom.
VINTREF	Output	Num	The vertical reference point of FIELD-NAME or FORM-NAME that the position is relative to. Values same as for VEXTREF.
ROW	Output	Num	The number of rows the field or form is offset from VREL-FIELDs reference point.
HREL-FIELD	Output	Str (120)	The qualified name of the field the form or field position is horizontally relative to or blank if it is relative to the containing form.
HEXTREF	Output	Num	The horizontal reference point of the field the form or field location is relative to. Values are: 1 - left, 2 - center, and 3 - right.
HINTREF	Output	Num	The horizontal reference point of FIELD-NAME or FORM-NAME that the position is relative to. Values the same as for HEXTREF.
COLUMN	Output	Num	The number of columns the field or form is offset from HREL-FIELD's reference point.

RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.36.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   VREL-FIELD           PIC X(120).
01   VEXTREF    PIC S9(5) COMP.
01   VINTREF    PIC S9(5) COMP.
01   ROW        PIC S9(5) COMP.
01   HREL-FIELD           PIC X(120).
01   HEXTREF    PIC S9(5) COMP.
01   HINTREF    PIC S9(5) COMP.
01   COLUMN     PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQLOC" USING EXPFRM-NAME,
               TSTFLD-NAME,
               VREL-FIELD,
               VEXTREF,
               VINTREF,
               ROW,
               HREL-FIELD,
               HEXTREF,
               HINTREF,
               COLUMN,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.37 INQPRO

This routine returns form or field prompt information. The output parameters that return position information correspond to the components of the Location syntax described in the Form Editor User's Manual.

4.37.1 Calling Format

```
CALL "INQPRO" USING      FORM-NAME,
                 FIELD-NAME,
                 PROMPT-NUM,
                 BUF-SIZE,
                 LENGTH,
                 TEXT,
                 VREL-FIELD,
                 VEXTREF,
                 VINTREF,
                 ROW,
                 HREL-FIELD,
                 HEXTREF,
                 HINTREF,
                 COLUMN,
                 RCODE.
```

4.37.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form or the name of the form whose prompt information you want.
FIELD-NAME	Input	Str(10)	The name of the field whose prompt information you want or blank if you want the prompt information for FORM-NAME.
PROMPT-NUM	Input	Num	A number based on the order the prompts were defined on the form or field to indicate which prompt you want the information about (i.e. a positive number of 1 or greater refers to the actual order prompts were defined; 0 refers to the most recently defined prompt; a negative 1 or greater refers to the reverse order prompts were defined starting with the next most recently added.
BUF-SIZE	Input	Num	The size of the buffer being reserved for the prompt text.
LENGTH	Output	Num	The actual length of the prompt text without padding. This is also the minimum buffer length needed to receive a non-truncated prompt text. Comparing LENGTH with BUF-SIZE determines whether TEXT is truncated or padded.
TEXT	Output	Str (BUF-SIZE)	The prompt text, either padded with spaces or truncated.

VREL-FIELD	Output	Str (120)	The qualified name of the field the prompt is vertically relative to or blank if it is relative to its containing form.
VEXTREF	Output	Num	The vertical reference point on VREL-FIELD that the prompt is relative to. Values are: 1 - top, 2 - center, and 3 - bottom.
VINTREF	Output	Num	The vertical reference point of the prompt that its position is relative to. Values same as for VEXTREF.
ROW	Output	Num	The number of rows the prompt is offset from VREL-FIELD's reference point.
HREL-FIELD	Output	Str (120)	The qualified name of the field the prompt is horizontally relative to or blank if it is relative to the containing form
HEXTREF	Output	Num	The horizontal reference point of the field the prompt is relative to. Values are: 1 - left, 2 - center, and 3 - right
HINTREF	Output	Num	The horizontal reference point of FIELD-NAME or FORM-NAME that the pprompt is relative to. Values the same as for HEXTREF.
COLUMN	Output	Num	The number of columns the prompt is offset from HREL-FIELD's reference point.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.37.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   PROMPT-NUM           PIC S9(5) COMP VALUE 1.
01   BUF-SIZE  PIC S9(5) COMP VALUE 100.
01   PUB-LEN   PIC S9(5) COMP.
01   TEXT-BUF  PIC X(100).
01   VREL-FIELD           PIC X(120).
01   VEXTREF   PIC S9(5) COMP.
01   VINTREF   PIC S9(5) COMP.
01   ROW       PIC S9(5) COMP.
01   HREL-FIELD           PIC X(120).
01   HEXTREF   PIC S9(5) COMP.
01   HINTREF   PIC S9(5) COMP.
01   COLUMN    PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQPRO" USING EXPFRM-NAME,
               TSTFLD-NAME,
               PROMPT-NUM,
               BUF-SIZE,
               BUF-LEN,
               TEXT-BUF,
               VREL-FIELD,
               VEXTREF,
               VINTREF,
               ROW,
               HREL-FIELD,
               HEXTREF,
               HINTREF,
               COLUMN,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.38 INQSIZ

This routine returns the size of a form or field. If the field is an array, the information is returned for an element of the array.

4.38.1 Calling Format

```
CALL "INQSIZ" USING      FORM-NAME,
                         FIELD-NAME,
                         WIDTH,
                         HEIGHT,
                         RCODE.
```

4.38.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form whose size you want or that contains the field whose size you want.
FIELD-NAME	Input	Str(10)	The name of the field whose size you want or blank if you want the size of FORM-NAME.
WIDTH	Output	Num	The number of columns wide the size is.
HEIGHT	Output	Num	The number of rows deep the size is.
RCODE	Output	Num	The routine return code. The possible values are defined in the include member FPCODE.

4.38.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   WIDTH      PIC S9(5) COMP.
01   HEIGHT     PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQSIZ" USING EXPFRM-NAME,
               TSTFLD-NAME,
               WIDTH,
               HEIGHT,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.39 INQTYP

This routine returns a field's type (i.e., either item, form, or window).

4.39.1 Calling Format

```
CALL "INQTYP" USING      FORM-NAME,
                         FIELD-NAME,
                         FIELD-TYPE,
                         RCODE.
```

4.39.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose type you want or blank if you just want to find out if FORM-NAME exists.
FIELD-TYPE	Output	Str(1)	A character code to identify the type of field FIELD-NAME is (W - window, F - form, and I - item). If blank, type will be F if FORM-NAME exists.
RCODE	Output	Num	The routine return code. The possible values are defined in the include member FPCODE.

4.39.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   TSTFLD-TYPE          PIC X.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQTYP" USING EXPFRM-NAME,
               TSTFLD-NAME,
               TSTFLD-TYPE,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.40 INQVAL

This routine returns the value expression defined for an item field. The EXPRESSION parameter corresponds to the Expression component of the VALUE clause syntax described in the Form Editor User's Manual.

4.40.1 Calling Format

```
CALL "INQVAL" USING      FORM-NAME,
                         FIELD-NAME,
                         BUF-SIZE,
                         EXPRESSION,
                         LENG
                         RCODE.
```

4.40.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose value expression you want.
BUF-SIZE	Input	Num	The length of the buffer being reserved for the value expression.
EXPRESSION	Output	Str (BUF-SIZE)	The value expression defined for the field either padded with spaces or truncated.
RCODE	Output	Num	The routine return code. The possible values are defined in the include member FPCODE.

This routine may first be called with a BUF-SIZE of zero. This forces an error but LENGTH will be the actual buffer size needed to store the value expression.

4.40.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   BUF-SIZE  PIC S9(5) COMP VALUE 100.
01   VALUE-BUF PIC X(100).
01   BUF-LEN   PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INQVAL" USING EXPFRM-NAME,
               TSTFLD-NAME,
               BUF-SIZE,
               VALUE-BUF,
               BUF-LEN,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.41 MAKFRM

This routine creates a complete form at run-time. Complete here means that no defaults are used. You must specify the size and background of the form. The form is open.

4.41.1 Calling Format

```
CALL "MAKFRM" USING      FORM-NAME,
                BACKGROUND,
                WIDTH,
                HEIGHT,
                RCODE.
```

4.41.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form to be created.
BACKGROUND	Input	Str(10)	The name of the attribute that specifies the form's background
WIDTH	Input	Num	The number of columns the form will occupy on its containing form.
HEIGHT	Input	Num	The number of rows the form will occupy on its containing form.
RCODE	Output	Num	The routine return code. The possible values are defined in the include member FPCODE.

4.41.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   NEWFRM-NAME          PIC X(10) VALUE "NEWFRM".
01   ZERO-COMP PIC S9(5) COMP VALUE ZERO.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "MAKFRM" USING NEWFRM-NAME,
               XPARENT,
               ZERO-COMP,
               ZERO-COMP,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.42 MOVLDV

This routine is used to move the specified logical device to a different physical device. For example, you may want to route your output from a screen to a specific line printer. You would specify the name of that printer as well as its type (port). The valid names for the logical and physical devices are set up by the system manager.

4.42.1 Calling Format

```
CALL "MOVLDV" USING      LDWNID,
                 DEVICE,
                 DEVTYP,
                 RCODE.
```

4.42.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
LDWNID	Input	Num	The id of the logical device that you want to move.
DEVICE	Input	Str(10)	The new device name which is a valid name according to the system manager.
DEVTYP	Input	Str(10)	The new device type which is a valid name according to the system manager.
RCODE	Output	Num	The routine return code. The possible values are defined in the include member FPCODE.

4.42.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   LDWNID                    PIC S9(5)   COMP.
01   DEVICE                    PIC X(10).
01   DEVTYP                    PIC X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "MOVLDV" USING LDWNID,
               DEVICE,
               DEVTYP,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.43 MVRFLD

This routine duplicates a field on all instances of another open form or on its containing form and positions it either absolutely or relatively as specified.

4.43.1 Calling Format

```
CALL "MVRFLD" USING      FORM-NAME,
                         FIELD-NAME,
                         COPY-FORM,
                         COPY-FIELD,
                         VREL-FIELD,
                         VEXTREF,
                         VINTREF,
                         ROW,
                         HREL-FIELD,
                         HEXTREF,
                         HINTREF,
                         COLUMN,
                         RCODE.
```

4.43.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field that you are copying.
COPY-FORM	Input	Str(10)	The name of the form that you are copying the field to. This may be the same as FORM-NAME.
COPY-FIELD	Input	Str(10)	The name of the field copy.

VREL-FIELD	Input	Str (120)	The qualified name of the field that the copied field position is vertically relative to or blank if it is relative to its containing form.
VEXTREF	Input	Num	The vertical reference point on VREL-FIELD that the position is relative to. Values are: 1 - top, 2 - center, and 3 - bottom.
VINTREF	Input	Num	The vertical reference point on the copied field that the position is relative to. Values same as for VEXTREF.
ROW	Input	Num	The number of rows the copied field's position is offset from VEXTREF's reference point.
HREL-FIELD	Output	Str (120)	The qualified name of the field that the copied field position is horizontally relative to or blank if it is relative to its containing form.
HEXTREF	Output	Num	The horizontal reference point on HREL-FIELD that the position is relative to. Values are: 1 - left, 2 - center, and 3 - right
HINTREF	Output	Num	The horizontal reference point on the copied field that the position is relative to. Values the same as for HEXTREF.
COLUMN	Output	Num	The number of columns the field is offset from HREL-FIELD's reference point.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.43.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   NEWFLD-NAME          PIC X(10) VALUE "NEWFLD".
01   REF-FLD NAME         PIC X(7) VALUE "TSTFLD;".
01   TOP-LEFT-REF         PIC S9(5) COMP VALUE 1.
01   CENTER-REF           PIC S9(5) COMP VALUE 2.
01   BOT-RIGHT-REF        PIC S9(5) COMP VALUE 3.
01   ZERO-COMP PIC S9(5) COMP VALUE 0.
01   TWO-COMP  PIC S9(5) COMP VALUE 2.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "MVRFLD" USING EXPFRM-NAME,
               TSTFLD-NAME,
               EXPFRM-NAME,
               NEWFLD-NAME,
               REF-FLD-NAME,
               BOT-RIGHT-REF,
               TOP-LEFT-REF,
               TWO-COMP,
               REF-FLD-NAME,
               CENTER-REF,
               CENTER-REF,
               ZERO-COMP,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.44 NUMELM

This routine returns the number of elements in an open-ended array.

4.44.1 Calling Format

```
CALL "NUMELM" USING      ARRAY-NAME,
                         ELEMENTS,
                         RCODE.
```

4.44.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
ARRAY-NAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the open-ended array.
ELEMENTS	Output	Num	The number of elements in the open-ended array.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.44.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   PATH-NAME PIC X(11) VALUE "EXPFRM.ARY;".
01   NUM-ELEM  PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "NUMELM" USING PATH-NAME,
                         NUM-ELEM,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.45 OISCR

This routine displays the current Display List on the user's terminal screen and accepts data from the user for a specified window. The user may only enter data in the items contained in the specified window. The application program is suspended until the user presses one of the function keys that the FP passes back to the application program. You may define the function keys PF5 through PF20 as you need for your application and the user will access the functions by toggling the <MODE> key until application mode appears in the mode item. The function keys PF0 through PF4 are pre-defined as <ENTER>, <MODE>, <MESSAGES>, <HELP>, and <QUIT>. The use of function keys is explained in more detail in the IISS Terminal Operator Guide.

4.45.1 Calling Format

```
CALL "OISCR" USING       QUALIFIED-WINDOW-NAME,
                 FUNCTION,
                 RCODE.
```

4.45.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-WINDOW-NAME	Input	Str (1-120)	The qualified name of the window that identifies which user entered data will be accepted for update.
FUNCTION	Output	Num	The number of the function key pressed when data entry was complete (e.g. for <PF5>, FUNCTION is 5).
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.45.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   FUNC       PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "OISCR" USING  SCREN,
                         FUNC,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.46 OPNFRM

This routine is used to retrieve a form definition. The form is made active and the default data is made available. If the form contains subforms they are also opened. All open forms are placed on an Open List associated with the current logical device. A form must then be put on the current Display List using ADDFRM before it can be displayed to the user. System performance can vary due to the overhead associated with form retrieval. When a form is opened, the form definition is brought into memory. This memory can be made available again by calling CLSFRM. Therefore, available memory and performance should both be considered when deciding whether to open and close forms as needed or open all forms at the beginning of the application. NOTE also that any form created at run-time is automatically open and run-time modifications are made to forms on the Open List associated with the current logical device and all instances of a form on the current Display List.

4.46.1 Calling Format

```
CALL "OPNFRM" USING      FORM-NAME,
                 RCODE.
```

4.46.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form you want to open.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.46.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "OPNFRM" USING EXPFRM-NAME,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.47 OPNLDV

This routine is used to open a new logical device within the application. This involves:

- o creating a new Open List
- o creating a new Display List
- o assigning the Display List a logical-device-id
- o returning the id to the application program

Creating a logical device does not destroy any previously created logical devices. An example of where you would want to open an additional logical device is the case of an interactive report writer where you need a logical device for input from the keyboard as well as one for output to a printer. The routine CHGLDV is used to control which logical device is the current Display List.

4.47.1 Calling Format

```
CALL "OPNLDV" USING      LOGICAL-DEVICE-ID,
                 RCODE.
```

4.47.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
LOGICAL-DEVICE-ID	Output	Num	The id of the logical device that you open.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.47.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   LOGICAL-DEVICE-ID   PIC S9(5)  COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "OPNLDV" USING LOGICAL-DEVICE-ID,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.48 OUTSCR

This routine displays the current Display List on the user's terminal screen. Control is returned immediately to the application program without allowing the user to enter any data. This routine can be used to let the user know that the application is "working" when a long operation is being performed.

4.48.1 Calling Format

```
CALL "OUTSCR" USING      QUALIFIED-WINDOW-NAME,
                 RCODE.
```

4.48.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-WINDOW-NAME	Input	Str (1-120)	Required, but not currently used.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.48.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "OUTSCR" USING SCREN,
                 RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.49 PARFQN

This routine lets you find out the name of the field and its type at a specified level of a specified fully qualified name. Calling this routine might be a natural action after calling GETCUR. You would be able to find out exactly where the action is happening.

4.49.1 Calling Format

```
CALL "PARFQN" USING      QUALIFIED-NAME,
                         FIELD-TYPE,
                         LEVEL,
                         LEVEL-NAME,
                         LEVEL-TYPE,
                         RCODE.
```

4.49.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-NAME	Input	Str (1-120)	The fully qualified name of the element in the Display List that you want to parse.
FIELD-TYPE	Input	Str(1)	A character code to identify the type of the last field in the fully qualified name. Values are: F - form, W - window, and I - item.
LEVEL	Input	Num	The level in the name that you want to identify. 1 - top level 0 - bottom level. Go down from the top with positive #s or up from the bottom with positive #
LEVEL-NAME	Output	Str (120)	The name of the field at the specified level of the fully qualified name.

LEVEL-TYPE	Output	Str(1)	A character code to identify the type of the parsed field. Values are: W - window, F - form and I - item.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.49.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   QUALIFIED-NAME       PIC X(120).
01   LEVEL-NAME           PIC X(120).
01   TYPE1      PIC X.
01   LEVEL-TYPE           PIC X.
01   ROW        PIC S9(5) COMP.
01   COL        PIC S9(5) COMP.
01   LEVEL      PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GETCUR" USING FULL-NAME,
               TYPE1,
               ROW,
               COL,
               RCODE.

     CALL "PARFQN" USING FULL-NAME,
               TYPE,
               LEVEL,
               LEVEL-NAME,
               LEVEL-TYPE,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.50 PDATA

This routine puts data in a Display List element defined by a qualified name. The specified element cannot be a window. This means that the data can only be put into a form or an item. Sufficient input data must be supplied to fill the Display List element. The program MAKINC can be used to generate a variable declaration which corresponds to the form definition.

4.50.1 Calling Format

```
CALL "PDATA" USING       QUALIFIED-NAME,
                 INPUT-DATA,
                 RCODE.
```

4.50.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-NAME	Input	Str (1-120)	The fully qualified name of the Display List element that is to receive the data. The routines GWINDO and GPAGE may have to be used to help you find this name
INPUT-DATA	Input	Var.	The data structure containing the data that you want to put into the element specified by qualified name. You have the option of defining the structure in your application or using the structure defined with the forms definition to MAKINC described in the Form Editor User Manual.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.50.3 Example

```
     WORKING-STORAGE SECTION.
     COPY FPCODE OF IISSCLIB.
     COPY FPPARM OF IISSCLIB.
     01   PATH-NAME PIC  X(7) value "EXPFRM;".
     COPY EXPFRM OF IISSCLIB.
    *EXPLANATION OF COPY MEMBER EXPFRM
    *01 EXPFRM.
    *   02 FIELD1   PIC X(6).
    *   02 FIELD2   PIC X(10).
    *   02 FIELD3   PIC XX.
     PROCEDURE DIVISION.
          .
          .
          .
          CALL "PDATA" USING  PATH-NAME,
                    EXPFRM,
                    RCODE.
          IF RCODE IS EQUAL TO OK
          .
          .
          .
          ELSE
             CALL "PMSGLC" USING RCODE.
```

4.51 PMSGLC

This routine matches a predetermined message with MCODE. It then inserts the message into the MESSAGE-LINE portion of the buffer so that the message will be displayed the next time the Display List is sent to the user's terminal. Follow this call with a call to OUTSCR or OISCR for immediate transmission. Return codes for the FP routines and their associated messages are defined in the include file FPCODE. Section 5.0 of this manual contains a complete list of FPCODE. The Message Management program was used to generate this file and can be used to generate programmer defined messages and codes. The Message Management program is documented in the IISS Terminal Operator Guide.

4.51.1 Calling Format

```
CALL "PMSGLC" USING      MCODE.
```

4.51.2 Parameter Descriptions

MCODE	Input	Str(5)	The code name of the message that will be displayed. This could be an RCODE defined in FPCODE or any other code name associated with a user created message in another include file

4.51.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
PROCEDURE DIVISION.
     .
     .
     .
     CALL"PMSGLC" USING MCODE.
```

4.52 PMSGLS

This routine inserts MSG-STRING into the MESSAGE-LINE portion of the buffer. The MESSAGE-LINE will be displayed the next time the Display List is sent to the user's terminal. Follow this call with a call to OUTSCR or OISCR for immediate transmission.

4.52.1 Calling Format

```
CALL "PMSGLS" USING      MSG-STRING.
```

4.52.2 Parameter Descriptions

MSG-STRING	Input	Str(60)	The character string that is to be displayed at the user's terminal.

4.52.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   MSG-STRING          PIC X(60) VALUE SPACES.
PROCEDURE DIVISION.
     .
     .
     .
     MOVE "YOU HAVE AN ERROR" TO MSG-STRING.
     CALL "PMSGLS" USING MSG-STRING.
```

4.53 PUTATT

The foreground attribute for items (specified during form definition with the DISPLAY AS clause) can be changed at run-time using this routine. The values specified during form definition are permanent. A foreground attribute can be changed permanently (i.e., for all following calls to OUTSCR or OISCR until PUTATT is called again) or temporarily (i.e., it overrides the current permanent attribute for the next call to OUTSCR or OISCR). When changed temporarily, the permanent value becomes effective again after the next call to OUTSCR or OISCR and the temporary value becomes blank. The routine GETATT allows you to find out the current values for either of the attribute types. Foreground attribute values that may only be defined temporarily are ERROR and GUARDED. ERROR highlights the item field in some manner (i.e., blinks). This is useful for showing a user that there is an error in a field value. GUARDED makes an item field not enterable.

4.53.1 Calling Format

```
CALL "PUTATT" USING      QUALIFIED-NAME,
                         ATTRIBUTE-TYPE,
                         ATTRIBUTE-ID,
                         RCODE.
```

4.53.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-NAME	Input	Str (1-120)	The fully qualified name of the Display List element that identifies the item(s) whose attributes you want to change. If element is a form or a window, all items contained in it will be changed.
ATTRIBUTE-TYPE	Input	Num	The attribute type (i.e. permanent or temporary) you want to change. The include file FPPARM contains the definitions for the type values TEMP and PERM.

ATTRIBUTE-ID	Input	Str(10)	The new attribute value to be associated with the specified item(s) for the specified attribute type. This can be a pre-defined value defined in the include file FPPARM, or a user-defined attribute.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.53.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   PATH-NAME PIC X(12) VALUE "FORM1.ITEM2;".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "PUTATT" USING PATH-NAME,
               TEMP,
               ERR,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.54 PUTBAK

The background of forms and windows (specified during form definition with the BACKGROUND clause) can be changed at run-time using this routine. The new attribute will remain in effect for all following calls to OUTSCR or OISCR until PUTBAK is called again. The routine GETBAK allows you to find out the current value of the background attribute for a form or window.

4.54.1 Calling Format

```
CALL "PUTBAK" USING      QUALIFIED-NAME,
                         BACKGROUND-TYPE,
                         BACKGROUND-ID,
                         RCODE.
```

4.54.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-NAME	Input	Str (1-120)	The fully qualified name of the Display List element that identifies the form or window whose background you want to change.
BACKGROUND TYPE	Input	Num	The background type (i.e. permanent or temporary) you want to change. The include file FPPARM contains the definitions for the type values TEMP and PERM. TEMP is not supported at this time.
BACKGROUND ID	Input	Str(10)	The new background value to be associated with the specified form or window for the speci- background type. This can be a pre-defined value defined in the include file FPPARM, or a user-defined attribute.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.54.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   PATH-NAME PIC X(12) VALUE "FORM1.ITEM2;".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "PUTBAK" USING PATH-NAME,
               TEMP,
               WHITE,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.55 PUTCUR

This routine allows you to specify where the cursor will be positioned the next time the Display List is sent to the user's terminal. The default cursor position is at the first input item if you do not use this routine before calling OISCR.

4.55.1 Calling Format

```
CALL "PUTCUR" USING      QUALIFIED-NAME,
                 RCODE.
```

4.55.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-NAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the field you want the cursor positioned in.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.55.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   PATH-NAME PIC X(12) VALUE "FORM1.ITEM2;".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "PUTCUR" USING PATH-NAME,
                 RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.56 PUTLOC

This routine allows you to specify where the cursor will be positioned within a field the next time the Display List is sent to the user's terminal. The default cursor position is at the first input item if you do not use this routine before calling OISCR.

4.56.1 Calling Format

```
CALL "PUTLOC" USING      QUALIFIED-NAME,
                 ROW,
                 COLUMN,
                 RCODE.
```

4.56.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-NAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the field you want the cursor positioned in.
ROW	Input	Num	The row within the field specified by the qualified name that you want the cursor to be positioned in.
COLUMN	Input	Num	The column within the field specified by the qualified name that you want the cursor to be positioned in.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.56.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   PATH-NAME PIC  X(12) VALUE "FORM1
01   ROW       PIC S9(5)  COMP VALUE 2.
01   COLUMN    PIC S9(5)  COMP VALUE 3.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "PUTLOC" USING "FORM1.ITEM2;",
               ROW,
               COLUMN,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.57 PUTVTI

This routine sends application data to the Virtual Terminal. The routine INITVT must be called first to enable VT pass-through mode.

4.57.1 Calling Format

```
CALL "PUTVTI" USING      BUFFER,
                         LENGTH,
                         RCODE.
```

4.57.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
BUFFER	Input	Str (LENG)	The data and commands being sent to the Virtual Terminal.
LENGTH	Input	Num	The actual size of BUFFER.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.57.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   VTI-BUF    PIC X  OCCURS 1 TO 1000 TIMES
                       DEPENDING ON BUF-LEN.
01   BUF-LEN    PIC S9(5) COMP.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INITVT" USING RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PUTVTI" USING  VTI-BUF,
                 BUF-LEN,
                 RCODE
        IF RCODE IS EQUAL TO OK
        .
        .
        .
        ELSE
           CALL "PMSGLC" USING RCODE.
```

4.58 REPFLD

This routine duplicates a field on all instances of another open form or on its containing form. The copied field has all the characteristics of the original field except its location. The field copy does not have a location and will not appear when its containing form is displayed unless the routine SETLOC is used to define a valid location for it.

4.58.1 Calling Format

```
CALL "REPFLD" USING      FORM-NAME,
                         FIELD-NAME,
                         COPY-FORM,
                         COPY-FIELD,
                         RCODE.
```

4.58.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field that you are copying.
COPY-FORM	Input	Str(10)	The name of the form that you are copying the field to. This may be the same as FORM-NAME.
COPY-FIELD	Input	Str(10)	The name of the field copy.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.58.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   NEWFLD-NAME          PIC X(10) VALUE "NEWFLD".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "REPFLD" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         EXPFRM-NAME,
                         NEWFLD-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.59 REPFRM

This routine duplicates an open form with a new name. The copied form has all the characteristics of the original form. NOTE that all instances of the form in the Display List are not copied.

4.59.1 Calling Format

```
CALL "REPFRM" USING      FORM-NAME,
                         COPY-NAME,
                         RCODE.
```

4.59.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form you are making a copy of.
COPY-NAME	Input	Str(10)	The name of the form copy.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.59.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   NEWFRM-NAME          PIC X(10) VALUE "NEWFRM".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "REPFRM" USING EXPFRM-NAME,
               NEWFRM-NAME,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.60 RMVAPR

This routine removes the criterion defined for specifying when a field appears on its containing form. This criterion corresponds to the Criterion component of the APPEARS IF clause syntax described in the Form Editor User's Manual.

4.60.1 Calling Format

```
CALL "RMVAPR" USING      FORM-NAME,
                 FIELD-NAME,
                 RCODE.
```

4.60.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose appears if criterion you want to remove.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.60.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVAPR" USING EXPFRM-NAME,
                 TSTFLD-NAME,
                 RCODE.
```

4.61 RMVARY

This routine removes all array dimensions from a field. This means there is only one occurrence of the field on its containing form.

4.61.1 Calling Format

```
CALL "RMVARY" USING      FORM-NAME,
                         FIELD-NAME,
                         RCODE.
```

4.61.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field you no longer want to be an array.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.61.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   ARYFLD-NAME          PIC X(10) VALUE "ARY".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVARY" USING EXPFRM-NAME,
               ARYFLD-NAME,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.62 RMVATT

This routine deletes one or all of the attributes defined for a form. Form attributes are defined with the ATTRIBUTE clause described in the Form Editor User's Manual.

4.62.1 Calling Format

```
CALL "RMVATT" USING      FORM-NAME,
                         ATTRIBUTE-ID,
                         RCODE.
```

4.62.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form whose attribute(s) you want to delete.
ATTRIBUTE-ID	Input	Str(10)	The name of the specific attribute you waant to delete or blank to delete all the form's attributes.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.62.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTATR-NAME          PIC X(10) VALUE "TESTATTR".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVATT" USING EXPFRM-NAME,
               TSTATR-NAME,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
      CALL "PMSGLC" USING RCODE.
```

4.63 RMVDIM

This routine removes a field's array dimension.

4.63.1 Calling Format

```
CALL "RMVDIM" USING      FORM-NAME,
                         FIELD-NAME,
                         DIM-INDEX,
                         RCODE.
```

4.63.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose array dimension you want to remove.
DIM-INDEX	Input	Num	A number to indicate which dimension to remove where 1 refers to the most inclusive, 2 the next most inclusive, and so on.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.63.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   ARYFLD-NAME          PIC X(10) VALUE "ARY".
01   TWO-COMP  PIC S9(5)  COMP VALUE 2.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVDIM" USING EXPFRM-NAME,
               ARYFLD-NAME,
               TWO-COMP,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.64 RMVDOM

This routine removes the domain options defined for an item field. Domain options correspond to DOMAIN clause options described in the Form Editor User's Manual.

4.64.1 Calling Format

```
CALL "RMVDOM" USING      FORM-NAME,
                         ITEM-NAME,
                         RCODE.
```

4.64.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
ITEM-NAME	Input	Str(10)	The name of the item field whose domain options you want to remove.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.64.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVDOM" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.65 RMVFLD

This routine deletes a specified field from an open form and all instances of the form on the current Display List.

4.65.1 Calling Format

```
CALL "RMVFLD" USING      FORM-NAME,
                         FIELD-NAME,
                         RCODE.
```

4.65.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field you want to delete.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.65.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVFLD" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.66 RMVHLP

This routine removes the help information defined for an item field.

4.66.1 Calling Format

```
CALL "RMVHLP" USING      FORM-NAME,
                         ITEM-NAME,
                         RCODE.
```

4.66.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
ITEM-NAME	Input	Str(10)	The name of the field whose help information you want to delete.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.66.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVHLP" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.67 RMVPAG

This routine changes the current Display List by removing pages from a window. If there are pages with higher page numbers than the page specified, they are also removed.

4.67.1 Calling Format

```
CALL "RMVPAG" USING      QUALIFIED-WINDOW-NAME,
                         PAGE-NUMBER,
                         RCODE.
```

4.67.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-WINDOW-NAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the window the page(s) will be removed from.
PAGE-NUMBER	Input	Num	The number identifying the latest page that will be removed from the window.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.67.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   PAGE-NUMBER          PIC S9(5) COMP VALUE 1.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVPAG" USING            SCREN,
               PAGE-NUMBER
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.68 RMVPRO

This routine removes prompts from a field or a form. If multiple prompts contain the same text, all such prompts are removed. You can also specify to remove all prompts from the form or field.

4.68.1 Calling Format

```
CALL "RMVPRO" USING      FORM-NAME,
                 FIELD-NAME,
                 LENGTH,
                 PROMPT-TEXT,
                 RCODE.
```

4.68.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form whose prompts you want to remove or that contains the field whose prompts you want to remove.
FIELD-NAME	Input	Str(10)	The name of the field whose prompts you want to remove or blank if you want to remove prompts from FORM-NAME.
LENGTH	Input	Num	The length of the prompt text to be removed to 0 to remove all the prompts on the form or field.
PROMPT-TEXT	Input	Str (LENG)	The actual prompt text to be removed. This parameter is ignored if LENGTH is 0.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.68.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   BLANK-NAME           PIC X(10) VALUE SPACES.
01   PROMPT-LEN           PIC S9(5) COMP VALUE 12.
01   PROMPT-TXT           PIC X(12) VALUE "EXAMPLE FORM".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVPRO" USING EXPFRM-NAME,
               BLANK-NAME,
               PROMPT-LEN,
               PROMPT-TXT,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.69 RMVVAL

This routine removes the value expression defined for an item field. This expression corresponds to the Expression component of the VALUE clause syntax described in the Form Editor User's Manual.

4.69.1 Calling Format

```
CALL "RMVVAL" USING      FORM-NAME,
                         FIELD-NAME,
                         RCODE.
```

4.69.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose value expression you want to delete.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.69.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RMVVAL" USING EXPFRM-NAME,
                         TSTFLD-NAME,
                         RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.70 RPLFRM

This routine allows you to change a specified page of a window on the current Display List by replacing the form currently on the page with a new form. The pages above and below the specified page remain unchanged.

4.70.1 Calling Format

```
CALL "RPLFRM" USING      QUALIFIED-WINDOW-NAME,
                         PAGE-NUMBER,
                         FORM-NAME,
                         RCODE.
```

4.70.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-WINDOW-NAME	Input	Str (1-120)	The qualified name of the Display List element that identifies the window that contains the page you want to change.
PAGE-NUMBER	Input	Num	The number identifying the page you want to change.
FORM-NAME	Input	Str(10)	The name of the new form that the page will contain.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.70.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   CURPAG     PIC S9(5) COMP VALUE 2.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "RPLFRM" USING SCREN,
               CURPAG,
               EXPFRM-NAME,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.71 SAVFRM

This routine saves a form that is created or modified at run-time. You can save the FDL source for the form, the compiled form or both. If the form already exists, you can also specify whether it should be overwritten. NOTE that the saved FDL source can be modified outside of the application.

4.71.1 Calling Format

```
CALL "SAVFRM" USING      FORM-NAME,
                         SAVE-MODE,
                         OVRIT-FLAG,
                         RCODE.
```

4.71.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form to be saved.
SAVE-MODE	Input	Str(1)	A character code to specify how to save the form. Values are: C - compiled form, S - FDL source, B - both.
OVRIT-FLAG	Input	Num	An overwrite flag. 0 - do not overwrite the source file if it already exists. 1 - overwrite any existing source file.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.71.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   BOTH-MODE PIC X VALUE "B".
01   OVERWRITE PIC S9(5) COMP VALUE 1.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SAVFRM" USING EXPFRM-NAME,
               BOTH-MODE,
               OVERWRITE,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.72 SETAPR

This routine allows you to define the criterion for when a field appears on a form. This criterion corresponds to the Criterion component in the APPEARS IF clause syntax described in the Form Editor User's Manual.

4.72.1 Calling Format

```
CALL "SETAPR" USING      FORM-NAME,
                         FIELD-NAME,
                         LENGTH,
                         CRITERION,
                         RCODE.
```

4.72.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
ITEM-NAME	Input	Str(10)	The name of the field whose appears if criterion you want to define.
LENGTH	Input	Num	The length of the string that contains the criterion.
CRITERION	Input	Str (LENG)	The string containing the criterion.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.72.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   CRIT-TXT  PIC X(12) VALUE "'TSTFLD' > 0".
01   CRIT-LEN  PIC S9(5) COMP VALUE 12.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETAPR" USING EXPFRM-NAME,
               TSTFLD-NAME,
               CRIT-LEN,
               CRIT-TXT,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.73 SETATT

This routine defines a new attribute for a form or modifies an existing one. The routine parameters correspond to the components of the ATTRIBUTE clause syntax described in the Form Editor User's Manual.

4.73.1 Calling Format

```
CALL "SETATT" USING      FORM-NAME,
                 ATTRIBUTE-ID,
                 BACK-COLOR,
                 DISPLAY-COLOR,
                 LENGTH,
                 PRIMITIVES,
                 RCODE.
```

4.73.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form whose attribute you want to define or modify.
ATTRIBUTE-ID	Input	Str(10)	The name of the attribute to define or modify.
BACK-COLOR	Input	Str(10)	The background color primitive or blank to not change it.
DISPLAY-COLOR	Input	Str(10)	The display color primitive or blank to not change or set it.
LENGTH	Input	Num	The length of the string containing the primitives, including separating characters of spaces or commas, or blank if the attribute only contains color primitives.

PRIMITIVES	Input	Str (LENG)	The attribute primitives other than color. Valid values are listed following the parameter descriptions.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

Valid attribute primitives are:

BOLD DIM
UNDERSCORE SLOWBLINK
FASTBLINK HIDDEN
GUARDED REVERSE
ORED NOWRITE
TABFIELD

These values and some pre-defined attribute-ids are described in the Form Editor User's Manual.

4.73.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTATR-NAME          PIC X(10) VALUE "TESTATTR".
01   PRIMITIVES           PIC X(12) VALUE "BOLD,GUARDED".
01   PRIM-LEN  PIC S9(5) COMP VALUE 12.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETATT" USING EXPFRM-NAME,
                 TSTATR-NAME,
                 YELLOW,
                 BLACK,
                 PRIM-LEN,
                 PRIMITIVES,
                 RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.74 SETDIM

This routine allows you to change how a field array dimension is defined. The routine parameters correspond to the components of the Repeat Spec syntax described in the Form Editor User's Manual.

4.74.1 Calling Format

```
CALL "SETDIM" USING      FORM-NAME,
                         FIELD-NAME,
                         DIM-INDEX,
                         DISPLAY-SIZE,
                         ACTUAL-SIZE,
                         DIRECTION,
                         SPACES,
                         RCODE.
```

4.74.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose array dimension you want to change.
DIM-INDEX	Input	Num	A number to indicate which array dimension you want to change. 1 - most inclusive dimension. 2 - the next most inclusive, and so on.
DISPLAY-SIZE	Input	Num	The number of array elements to be displayed. 0 - open-ended.
ACTUAL-SIZE	Input	Num	The actual number of array elements there are. 0 - open-ended.

DIRECTION	Input	Str(1)	A character code to specify whether the array elements repeat vertically (V) or horizontally (H).
SPACES	Input	Num	The number of spaces to leave between repetitions.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.74.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFRD".
01   DIM-INDEX PIC S9(5)  COMP VALUE 1.
01   DSP-SIZE  PIC S9(5)  COMP VALUE 10.
01   ACT-SIZE  PIC S9(5)  COMP VALUE 20.
01   NUM-SPACES           PIC S9(5) COMP VALUE 0.
01   RPT-DIRECTION        PIC X VALUE "V".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETDIM" USING            EXPFRM-NAME,
               TSTFLD-NAME,
               DIM-INDEX,
               DSP-SIZE,
               ACT-SIZE,
               NUM-SPACES,
               RPT-DIRECTION,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
      CALL "PMSGLC" USING RCODE.
```

4.75 SETDIS

This routine defines or changes the display attribute for a field. This attribute corresponds to the Attribute_id component in the DISPLAY AS clause syntax described in the Form Editor User's Manual.

4.75.1 Calling Format

```
CALL "SETDIS" USING      FORM-NAME,
                         FIELD-NAME,
                         ATTRIBUTE-ID,
                         RCODE.
```

4.75.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose display attribute you want to change or define.
ATTRIBUTE-ID	Input	Str(10)	The name of the display attribute. This value must be previously defined for the form using SETATT or one of the valid pre-defined attributes contained in the include file FPPARM.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.75.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE FROM IISSCLIB.
COPY FPPARM FROM IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETDIS" USING            EXPFRM-NAME,
               TSTFLD-NAME,
               OUTP,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.76 SETDOM

This routine defines the domain options for an item field. These options correspond to the options available in the DOMAIN clause syntax described in the Form Editor User's Manual.

4.76.1 Calling Format

```
CALL "SETDOM" USING      FORM-NAME,
                         ITEM-NAME,
                         LENGTH,
                         DOMAIN,
                         RCODE.
```

4.76.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
ITEM-NAME	Input	Str(10)	The name of the item whose domain you want to define.
LENGTH	Input	Num	The length of the string that contains the domain options including separating commas.
DOMAIN	Input	Str (LENG)	The string containing the domain options separated by commas
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

Valid domain options are:

LEFT
RIGHT
UPPER
LOWER
MUST ENTER
MUST FILL
NUMERIC
MAXIMUM int
MINIMUM int

4.76.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   DOMAIN     PIC X(6) VALUE "NUMERIC".
01   DOMAIN-LEN           PIC S9(5) COMP VALUE 6.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETDOM" USING EXPFRM-NAME,
               TSTFLD-NAME,
               DOMAIN-LEN,
               DOMAIN,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.77 SETDQN

This routine sets a default starting place for identifying elements in the current Display List with a relative qualified name. Initially, this starting place is the form PSCREN at the top of the Display List. All further Form Processor relative qualified names will be relative to this field until the default is reset or the field is deleted.

4.77.1 Calling Format

```
CALL "SETDQN" USING      QUALIFIED-NAME,
                         RCODE.
```

4.77.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
QUALIFIED-NAME	Input	Str (1-120)	The qualified name that identifies the default starting place in the Display List for relative qualified names.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.77.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   PATH-NAME PIC X(13) VALUE "WINDOW.FORM1;".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETDQN" USING            PATH-NAME,
                 RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.78 SETHLP

This routine defines or changes the help information available for an item field. The routine parameters correspond to the components of the HELP clause syntax described in the Form Editor User's Manual.

4.78.1 Calling Format

```
CALL "SETHLP" USING      FORM-NAME,
                         ITEM-NAME,
                         TYPE,
                         HELP,
                         RCODE.
```

4.78.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
ITEM-NAME	Input	Str(10)	The name of the item field whose help information you want to change or define.
TYPE	Input	Str(10)	A character to specify the type of help being defined. Values are: A - application, F - form, and S - string.
HELP	Input	Str (1-60)	The help string being defined. Its contents depend on the TYPE value. When TYPE = F, HELP contains the name of a help form; if TYPE = A it is ignored; and if TYPE = S HELP contains the actual help string.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.78.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   HELP-TYPE PIC X(1)   VALUE "S".
```

```
01   HELP-MSG  PIC X(60)
                    VALUE "SAMPLE HELP MESSAGE".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETHLP" USING EXPFRM-NAME,
               TSTFLD-NAME,
               HELP-TYPE,
               HELP-MSG,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.79 SETLDV

This routine sets display parameters (row, col, width and depth) for a logical device belonging to an application. If the logical device is not opened SETLDV will return an NFPDSTRC error.

4.79.1 Calling Format

```
CALL "SETLDV" USING      LDWNID,
                 DSPROW,
                 DSPCOL,
                 DSPWDTH,
                 DSPDPTH,
                 RCODE.
```

4.79.2 Parameter Description

NAME	I/O	FORMAT	DEFINITION
LDWNID	Input	Num	The id of the logical device
DSPROW	Input	Num	The row position of the origin of the logical device display.
DSPCOL	Input	Num	The column position of the origin of the logical device display.
DSPWDTH	Input	Num	The width of the logical device's display.
DSPDPTH	Input	Num	The depth of the logical device's display.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.79.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETLDV" USING LDWNID,
               DSPROW,
               DSPCOL,
               DSPWDTH,
               DSPDPTH,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.80 SETLOC

This routine defines the position of a field on a form. The routine parameters correspond to the components of the Location syntax described in the Form Editor User's Manual.

4.80.1 Calling Format

```
CALL "SETLOC" USING      FORM-NAME,
                         FIELD-NAME,
                         VREL-FIEL,
                         VEXTREF,
                         VINTREF,
                         ROW,
                         HREL-FIELD,
                         HEXTREF,
                         HINTREF,
                         COLUMN,
                         RCODE.
```

4.80.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose position you want to define.

VREL-FIELD	Input	Str (120)	The qualified name of the field that FIELD-NAME's position is vertically relative to or blank if it is relative to its containing form.
VEXTREF	Input	Num	The vertical reference point on VREL-FIELD that FIELD-NAME's position is relative to. Values are: 1 - top, 2 - center, and 3 - bottom.
VINTREF	Input	Num	The vertical reference point of FIELD-NAME that its position is relative to. Values are the same as for VEXTREF.
ROW	Input	Num	The number of rows that FIELD-NAME's position is offset from VREL-FIELD's reference point.
HREL-FIELD	Input	Str (1-120)	The qualified name of the field that FIELD-NAME's position is horizontally relative to or blank if it is relative to its containing form.
HEXTREF	Input	Num	The horizontal reference point on HREL-FIELD that FIELD-NAME's position is relative to. Values are: 1 - left, 2 - center, and 3 - right.
HINTREF	Input	Num	The horizontal reference point of FIELD-NAME that the position is relative to. Values the same as for HEXTREF.
COLUMN	Input	Num	The number of columns the field is offset from HREL-FIELD's reference point.

RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.80.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   NEWFLD-NAME          PIC X(10) VALUE "NEWITEM".
01   REF-FLD-NAME         PIC X(7) VALUE "TSTFLD;".
01   TOP-LEFT-REF         PIC S9(5) COMP VALUE 1.
01   CENTER-REF           PIC S9(5) COMP VALUE 2.
01   BOT-RIGHT-REF        PIC S9(5) COMP VALUE 3.
01   ZERO-COMP PIC S9(5) COMP VALUE 0.
01   TWO-COMP  PIC S9(5) COMP VALUE 2.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETLOC" USING EXPFRM-NAME,
               NEWFLD-NAME,
               REF-FLD-NAME,
               BOT-RIGHT-REF,
               TOP-RIGHT-REF,
               TWO-COMP,
               REF-FLD-NAME,
               CENTER-REF,
               CENTER-REF,
               ZERO-COMP,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.81 SETNAM

This routine renames a field on a form.

4.81.1 Calling Format

```
CALL "SETNAM" USING      FORM-NAME,
                         OLD-NAME,
                         NEW-NAME,
                         RCODE.
```

4.81.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
OLD-NAME	Input	Str(10)	The current name of the field.
NEW-NAME	Input	Str(10)	The new name you want the field to have.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.81.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   NEWFLD-NAME          PIC X(10) VALUE "NEWFLD".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETNAM" USING EXPFRM-NAME,
               TSTFLD-NAME,
               NEWFLD-NAME,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.82 SETPRO

This routine adds a prompt to a form or a field at a specified position. The routine parameters that specify the prompt position correspond to the components of the Location syntax described in the Form Editor User's Manual.

4.82.1 Calling Format

```
CALL "SETPRO" USING      FORM-NAME,
                         FIELD-NAME,
                         LENGTH,
                         TEXT,
                         VREL-FIELD,
                         VEXTREF,
                         VINTREF,
                         ROW,
                         HREL-FIELD,
                         HEXTREF,
                         HINTREF,
                         COLUMN,
                         RCODE.
```

4.82.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the form to add the prompt to or the containing form.
FIELD-NAME	Input	Str(10)	The name of the field to add the prompt to or blank if the prompt is being added to FORM-NAME.
LENGTH	Input	Num	The actual length of the prompt text to be added.
TEXT	Input	Str (LENG)	The prompt text to be added.

VREL-FIELD	Input	Str (1-120)	The qualified name of the field that the prompt is vertically relative to or blank if it is relative to its containing form
VEXTREF	Input	Num	The vertical reference point on VREL-FIELD that the prompt position is relative to. Values are: 1 - top, 2 - center, and 3 - bottom.
VINTREF	Input	Num	The vertical reference point of the prompt that its position is relative to. Values are the same as for VEXTREF.
ROW	Input	Num	The number of rows the prompt's position is offset from VREL-FIELD's reference point.
HREL-FIELD	Input	Str (1-120)	The qualified name of the field that the prompt's position is horizontally relative to or blank if it is relative to its containing form.
HEXTREF	Input	Num	The horizontal reference point of the prompt that its position is relative to. Values are: 1 - left, 2 - center, and 3 - right.
HINTREF	Input	Num	The horizontal reference point of the prompt that the position is relative to. Values the same as for HEXTREF.
COLUMN	Input	Num	The number of columns the prompt is offset from HREL-FIELD's reference point.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.82.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   PROMPT-TEXT          PIC X(10) VALUE "NEW PROMPT".
01   PROMPT-LEN           PIC S9(5) COMP VALUE 10.
01   REFFLD-NAME          PIC X(7) VALUE "TSTFLD:'.
01   TOP-LEF-REF          PIC S9(5) COMP VALUE 1.
01   CENTER-REF           PIC S9(5) COMP VALUE 2.
01   BOT-RIGHT-REF        PIC S9(5) COMP VALUE 3.
01   ZERO-COMP PIC S9(5) COMP VALUE 0.
01   TWO-COMP  PIC S9(5) COMP VALUE 2.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETPRO" USING EXPFRM-NAME,
               TSTFLD-NAME,
               PROMPT-LEN,
               PROMPT-TEXT,
               REFFLD-NAME,
               TOP-LEFT-REF,
               BOT-RIGHT-REF,
               TWO-COMP,
               REFFLD-NAME,
               DENTER-REF,
               ZERO-COMP,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.83 SETSIZ

This routine allows you to specify the size of a field.

4.83.1 Calling Format

```
CALL "SETSIZ" USING      FORM-NAME,
                         FIELD-NAME,
                         WIDTH,
                         HEIGHT,
                         RCODE.
```

4.83.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose size you want to specify.
WIDTH	Input	Num	The number of columns wide the field is.
HEIGHT	Input	Num	The number of rows high the field is.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.83.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   NEW-WIDTH PIC S9(5) COMP VALUE 20.
01   NEW-HEIGHT           PIC S9(5) COMP VALUE 1.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETSIZ" USING          EXPFRM-NAME,
               TSTFLD-NAME,
               NEW-WIDTH,
               NEW-HEIGHT,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.84 SETTYP

This routine allows you to change a field's type on all instances of an open form.

4.84.1 Calling Format

```
CALL "SETTYP" USING      FORM-NAME,
                         FIELD-NAME,
                         NEW-TYPE,
                         RCODE.
```

4.84.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
FIELD-NAME	Input	Str(10)	The name of the field whose type you want to change.
NEW-TYPE	Input	Str(1)	A character code to specify the field's new type. Values are: W - window, F - form, and I - item.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.84.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   NEW-TYPE  PIC X VALUE "W".
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETTYP" USING EXPFRM-NAME,
               TSTFLD-NAME,
               NEW-TYPE,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.85 SETVAL

This routine allows you to define a value expression for an item field. This expression corresponds to the Expression component in the VALUE clause syntax described in the Form Editor User's Manual.

4.85.1 Calling Format

```
CALL "SETVAL" USING      FORM-NAME,
                         ITEM-NAME,
                         LENGTH,
                         EXPRESSION,
                         RCODE.
```

4.85.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
FORM-NAME	Input	Str(10)	The name of the containing form
ITEM-NAME	Input	Str(10)	The name of the item field whose value expression you want to define.
LENGTH	Input	Num	The length of the string that contains the value expression.
EXPRESSION	Input	Str (LENG)	The string containing the value expression.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.85.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01   EXPFRM-NAME          PIC X(10) VALUE "EXPFRM".
01   TSTFLD-NAME          PIC X(10) VALUE "TSTFLD".
01   VALUE-TXT.
     05 FILLER PIC X      VALUE QUOTE.
     05 FILLER PIC X(13)  VALUE "INITIAL VALUE".
     05 FILLER            PIC X     VALUE QUOTE.
01   VALUE-LEN PIC S9(5) COMP VALUE 15.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "SETVAL" USING EXPFRM-NAME,
               TSTFLD-NAME,
               VALUE-LEN,
               VALUE-TXT,
               RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

4.86 TERMFP

This routine is used to exit the FP. TERMFP must be called to return terminal characteristics to the settings they were at before entering the FP.

4.86.1 Calling Format

CALL "TERMFP".

4.86.2 Parameter Descriptions

NONE.

4.86.3 Example

```
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
     CALL "INITFP".
     .
     .
     .
     CALL "TERMFP".
EXIT-PROGRAM.
```

4.87 TERMVT

This routine clears the VTI mode flag.

4.87.1 Calling Format

```
CALL "TERMVT" USING      RCODE.
```

4.87.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

4.87.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INITVT" USING RCODE.
     IF RCODE IS EQUAL TO OK
     .
     .
     .
        CALL "TERMVT" USING RCODE
        IF RCODE IS EQUAL TO OK
           NEXT SENTENCE
        ELSE
           CALL "PMSGLC" USING RCODE
     ELSE
        CALL "PMSGLC" USING RCODE.
```

SECTION 5

INCLUDE FILES

The FP routines use the include files FPPARM and FPCODE. The symbol IISSCLIB points to the location containing these files. The following sections list the information defined in the files.

5.1 FPPARM

The values for various constant data fields used by the FP routines are initialized in this include file. These data fields are:

Instances

62;1;2;6;7;8;9c CURRNT - The value to identify the current instance of user entered data.

PREV - The value to identify the previous instance of user entered data.

Window Names

SCREN - The value to identify the top window of the Display List.

Foreground Attributes

INP - The value for the INPUT attribute that specifies that the user may enter data for this item and the value will be echoed on the screen. The user may tab to this item.

OUTP - The value for the OUTPUT attribute that specifies that only the application program may enter data for this item.

HIDDEN - The value for the HIDDEN attribute that specifies that the user may enter data for this item but the value will not be echoed on the screen.

ERR - The value for the ERROR attribute that causes the item to be highlighted when displayed on the screen (e.g., the item will blink). The user may enter data for this item and the value will be echoed on the screen. This attribute may only be set as a temporary attribute. It is useful for showing the user which items have errors in them.

GUARDED - The value for the GUARDED attribute that specifies that only the application program may enter a value for this item. The value is displayed in bold on terminals that support bold.

TXT - The value for the TEXT attribute that specifies that only the application program may enter a value for this item. The value is not in bold.

CALC - The value for the CALCULATED attribute that specifies that only the Form Processor may enter a value for this item. The value is displayed in bold.

TBFLD - The value for the TABFLD attribute that specifies that only the application program may enter a value for this item but the user may tab to the item.

Function Keys

HELPKEY - The number of the terminal control key that the user can press to display additional information or an explanation of the data to be entered for a form or form field.

QUITKEY - The number of the terminal control key that the user can press to end data entry and bypass edit checks.

ENTERKEY - The number of the terminal control key that the user can press to end data entry and return control to the application.

LSCRLKEY - The number of the terminal control key that the user can press in "scrll/page" mode to horizontally scroll the data on the screen to the left.

RSCRLKEY - The number of the terminal control key that the user can press in "scrll/page" mode to horizontally scroll the data on the screen to the right.

USCRLKEY - The number of the terminal control key that the user can press in "scrll/page" mode to vertically scroll the data on the screen up.

DSCRLKEY - The number of the terminal control key that the user can press in "scrll/page" mode to vertically scroll the data on the screen down.

LPAGEKEY - The number of the terminal control key that the user can press in "scrll/page" mode to horizontally page the data on the screen to the left.

RPAGEKEY - The number of the terminal control key that the user can press in "scrll/page" mode to horizontally page the data on the screen to the right.

UPAGEKEY - The number of the terminal control key that the user can press in "scrll/page" mode to vertically page the data on the screen up.

DPAGEKEY - The number of the terminal control key that the user can press in "scrll/page" mode to vertically page the data on the screen down.

Background Attributes

BLACK - The attribute value for: black background, white foreground.

WHITE - The attribute value for: white background, white foreground.

RED - The attribute value for: red background, white foreground.

GREEN - The attribute value for: green background, white foreground.

YELLOW - The attribute value for: yellow background, white foreground.

BLUE - The attribute value for: blue background, white foreground.

MAGENTA - The attribute value for: magenta background, white foreground.

CYAN - The attribute value for: cyan background, white foreground.

Duration Attributes

TEMP - The value for the attribute type "temporary".

PERM - The value for the attribute type "permanent".

NOTE: The FP function keys mentioned above and the <ENTER> key are terminal specific. The application user will have to be given this information based on the type of terminal available as explained in the IISS Terminal Operator Guide.

5.2 FPCODE

This include file contains the routine return code values. These return codes indicate irregular use of the FP routines. An application can display messages related to the codes in the message line by using the FP routines PMSGLC and PMSGLC. A developer testing a new application may want to use the debug option to display these messages without calling PMSGLC. The debug option is explained in the Terminal Operator Guide. The code names and their related messages are:

Number	Name	Message
00000	OK	(Blank - normal completion)
00001	ALCERR	Memory Allocation Error
02000	FILOERR	File open error

02001	FILNTFND	File not found
02002	FILNTCRT	File not created
02003	FILNTDEL	File not deleted
02004	ENDOFFIL	End of file
02005	FILNTOPN	File not opened
02006	FILNTCLS	File not closed
02007	FILRERR	File read error
02008	FILWERR	File write error
02009	FILNTIND	File not indexed
02010	FILNTSEQ	File not sequential
07000		MATH ERRORS
07001	INTOVRFL	Integer overflow
07002	FLTOVRFL	Floating point overflow
07003	ZERODIV	Zero Divide
70300		FORM PROCESSOR MESSAGES
70301	INVPAG	Invalid page number
70302	FNOTFND	Form not found
70303	FISOPEN	Form is already open
70304	FRMCYCLE	Cyclic reference in form name
70305	OPNERR	Open error - Unable to read form definition file
70306	EXPERR	Error in expression
70307	TRNCFLD	Field value too long - truncated
70308	UNKTYPE	Internal error - Unknown field type
70309	PNOTARY	Qualified name is not an array
70310	NILKEY	Non-functional key
70311	SYSERR	System error - Call System Administrator
70312	PBFULL	Paste buffer full
70313	INVEDT	Field can not be edited
70314	INVISIZ	Items must be fixed in size
70315	PATHERR	Invalid path name
70316	NOMACH	String not found
70317	PNOTFND	Path not found
70318	WRDWRAP	Word too long to fit between fill margins - wrapped
70319	INVMRG	Invalid fill margins
70320	PNOTUNQ	Path not unique
70321	PNOTWIN	Qualified name is not a window
70322	FNOTOPN	Form is not open
70323	NOHELP	No help available
70324	NOHLPFRM	Error opening help form
70325	BLKFLD	Field must be entered
70326	IMBBLK	Field must not contain blanks
70327	NFLTDATA	Field must contain only real numbers
70328	NINTDATA	Field must contain only integer numbers
70329	OUTRNG	Field value is out of range
70330	ATNOTFND	Attribute does not exist
70331	FINUSE	Can not close form - Form is in use
70332	MFOPNERR	Message file can not be opened
70333	INVKEY	Requested function can not be performed on this screen
70334	INVIID	Invalid instance ID
70335	NTMREJ	Network Transaction Manager (NTM) reject
70336	MAPFAIL	Virtual terminal buffer format error
70337	BUFOVFL	Virtual terminal buffer overflow
70338	GTCURERR	GETCUR could not build fully qualified name
70339	PARSERR	Level specified does not exist

70340	PNOTITEM	Qualified name not an item
70341	OLDFORM	Old form definition file format - Recompile
70342	ENDARY	End of scrolling section reached
70343	NOTSCROL	Field is not scrollable
70344	PNOTBACK	Qualified name not a window or form
70345	FILFRMMM	Form name does not match form definition file name
70346	APKEY	Function key is to be processed by application
70347	INTADDER	Internal error - Unable to add form to window
70348	INDEB	In debug mode
70349	OUTDEB	Out of debug mode
70350	MSGRANGE	Message number specified does not exist
70351	INVRPT	Invalid repeat count
70352	BADDEV	Unable to access specified device
70353	BADAP	Unable to start specified application
70354	BADAPS	Unable to create application data structures
70355	INVLDIM	Invalid dimension
70356	BFOVRFLW	Data will not fit into buffer provided
70357	BFUNDFLW	Not enough data to fill the buffer provided
70358	INTADLER	Internal Error - Could not add element to open-ended array
70359	INTGETER	Internal error - Unable to get data from forms
70360	NTRMVER	Internal error - Unable to remove form from window
70361	OLNCHG	Role not changed
70362	WNDNSEL	Window not selected
70363	UISWNNF	UIS window not found
70364	SDPNDNG	Shut down pending
70365	SDCNCLD	Shut down cancelled
70366	INVRLFN	Invalid role/function
70367	NFPDSTRC	No FPD Structure found
70368	NWNDIDFN	No unused window ID found
70369	BADFPD	Bad logical device data structure - (Bad FPD)
70370	WNDSEL	Window is selected
70371	WMARCHSZ	Cannot change size of window which is an element in an array
70372	WMARCHLC	Cannot change pos. of window which is an element in an array
70373	SCPWNOPN	Script file could not be opened for writing
70374	SCPRNOPN	Script file could not be opened for reading
70375	SCPSNOPN	Script output save file could not be opened for writing
70376	SCPBADFL	Script file is bad
70377	SCPSVERR	Could not write to script output save file - close save file
70378	SCPWRERR	Could not write to script file - terminating scripting
70379	SCPRDERR	Could not read script file - terminating scripting session
70380	OVRFLW	Adding an element would exceed bounds of form
70381	BADDATA	Non-printable characters in data - replaced with "?"
70382	WNDUNSEL	Window is unselected
70383	APNOTFND	Application not found

70384	CURFPDST	Cannot remove or close current logical device standalone
70385	NINVTIMD	Not in Form Processor bypass mode (VTI mode)
70386	INVTIMD	In Form Processor bypass mode (VTI mode)
70387	INTPUTER	Internal Error - Could not put data into field
70388	INTPATER	Internal Error - Could not charge attribute for a field
70389	GTNAMERR	Could not find name
70390	GTDQNERR	Could not find default qualified name
70391	AIVERORR	Wrong version of Application Interface - Relink application
70392	NODEPVAL	No dependent value clause found
70393	VALNTFND	Value clause not found
70394	FUNCNQ	Function is not unique
70395	NOFNCDEF	No definition of function in UI Database
70386	ILTYPCNV	Illegal type conversion in value clause
70397	ILLGLNAM	Illegal Name
70398	DUPLCNAM	Duplicate Name
70399	FLDNTFND	Field Not Found
70400		FORM PROCESSOR CODES (CONTINUED)
70401	ILLGLRFP	Illegal reference point
70402	FLDNTARR	Field is not an array
70403	INVALSEL	Invalid command selection
70404	NOMORFLD	End of field list
70405	INVINDEX	Invalid index specified
70406	MINMAXER	Minimum cannot be greater than maximum
70407	DOMNTFND	Domain value not found
70408	PMTNFND	Prompt not found
70409	ARNOTFND	Array not found
70410	HCYREF	Horizontal cyclical reference in field location
70411	VCYREF	Vertical cyclical reference in field location
70412	iNOHREF	Horizontally referenced field not found
70413	NOVREF	Vertically referenced field not found
70414	OFFRIGHT	Field's location is off right of form
70415	OFFLEFT	Field's location is off left of form
70416	OFFTOP	Field's location is off top of form
70417	OFFBOTTM	Field's location is off bottom of form
70418	OVLPFLDS	Two or more fields or prompts overlap
70419	TOONAROW	Fields extend off form to the right
70420	TOOSHORT	Fields extend below form
70421	VALDEP	Other fields depend on this field for their value
70422	INVHLPTP	Invalid help type - must be: (A)pplication, (F)orm, (S)tring
70423	FMTRUFLS	Flag must be (T)rue or (F)alse
70424	DOMNTCHR	Item's domain type not character
70425	DOMNTINT	Item's domain type not integer
70426	DOMNTFLD	Item's domain type not floating point
70427	PRMNTFND	Attribute primitive not found
70428	FRGNTFND	Foreground color not found
70429	BKGNTFND	Background color not found
70430	PRVFDFIL	FD file already exists : specify replace
70431	PRVFDLFL	FDL file already exists : specify replace
70432	NODEVICE	The current logical device has no physical device
70433	NOTSLIN	Field is not a non-graphics line

70434	NOLSTYL	No display style defined for non-graphics line
70435	STYTRNC	Non-graphics line display style truncated
70436	ILLGLLEN	Invalid length for non-graphics line display style
70437	PNOTFRM	Qualified name is not a form
70438	BDOFFSET	Window offsets must be positive numbers
79900		ERRORS THAT ARE LOGGED
79901	NTMINT	Bad return from NETWORK TRANSACTION MANAGER (NTM) INITEX routine
79902	NTMLOG	Bad return from NTM Logon routine
79903	NTMLO	Bad return from NTM Logoff routine
79904	BSEND	Bad send from UI to AP
79905	CROLER	Bad return from NTM CHGROL routine
79906	UOPFRM	Bad return from open form routine
79907	NTMMSGNA	NTM message not accepted
79908	NTMPRCNT	NTM terminate process message not accepted
79909	FPABORT	Form Processor ABORT
79910	USRABRT	User aborted
79911	FLDTBIG	Field too big to be sent by the NTM and was truncated

SECTION 6

ACCESSING THE FP ROUTINES

The following describes the environment required for running the IISS User Interface (UI) on a DEC VAX under the VMS operating system.

6.1 Logicals

The following are the logical symbols required.

IISSCLIB - Points to the IISS include library which contains all the COBOL include files.

IISSFLIB - Points to the directory which the Form Processor searches second for FD files (form definition). It must be defined at the JOB level.

IISSILIB - Points to the directory which contains the source for COBOL include files.

IISSMLIB - Points to the Form Processor's message files. It must be defined at the JOB level.

IISSSLIB - Points to the directory which contains Form Definition Language (FDL) source files. It must be defined at the JOB level.

IISSULIB - Points to the directory which the Form Processor searches first for FD (form definition) files. It must be defined at the JOB level.

6.2 Linking

The following are the object libraries for linking with the Form Processor and Virtual Terminal.

The object libraries required for an application to run under IISS are:

```
uidir:[util]utilolb/i=sysmsg
cdmdir:[fpai]fpaiolb/lib
cdmdir:[cdmr]cdmolb/lib
ntmdir:[services]servolb/lib
uidir:[util]utilolb/lib
uidir:[clib]clibolb/lib
uidir:[ipc]ipcolb/lib
sys$library:vaxcrtl/lib
```

Note: The CDM library is not required if the CDM is not used.

For development and testing purposes a version f the NTM with limited capabilities is available. It's link is as follows:

```
uidir:[minintm]miniolb/lib
uidir:[fpai]fpaiolb/lib
uidir:[util]utilolb/l
uidir:[clib]clibolb/lib
uidir:[ipc]ipcolb/lib
sys$library:vaxcrtl/lib
```

For development and testing purposes there is a version of the Form Processor that runs as a standalone operation for applications which do not require the NTM. Use the following object libraries:

```
uidir:[safp]safpolb/lib
uidir:[util]utilolb/l/i=(sysmsg,fndmsg)
uidir:[fp]fpolb/lib/i=prnfld
uidir:[driver]drivolb/lib/inc=(dddd,prnwnd)
uidir:[test]testolb/lib
uidir:[clib]clibolb/lib
sys$library:vaxcrtl/lib
```

where 'dddd' is the appropriate Device Driver (i.e., VT100, TEK4100, or IBM3270).

SECTION 7

SAMPLE APPLICATION PROGRAM

This is the "C" source code for the Message Management program which uses the FP routines. Figure 7-1 shows the form that the user interfaces with.

```
+-------------------------------------------------------------+
|              ERROR MESSAGE DEFINITION SCREEN                |
|              ----- ------- ---------- ------                |
|                                                             |
| Message Base Number:                                        |
|                      -------                                |
| NUMBER      NAME     DESCRIPTION                            |
|             -------- ----------------------------------      |
|             -------- ----------------------------------      |
|             -------- ----------------------------------      |
|             -------- ----------------------------------      |
|             -------- ----------------------------------      |
|             -------- ----------------------------------      |
|             -------- ----------------------------------      |
|             -------- ----------------------------------      |
|             -------- ----------------------------------      |
|             -------- ----------------------------------      |
|                                                             |
| MSG: 0                                          Applcation  |
+------------------  ------------------------------------------+
```

Figure 7-1 Sample Form

```
/* NAME
 *    MM - Message Management
 *       Written: 14-FEB-1984 14:24:47
 *       Revised: 17-JUL-1985 10:54:01 - JONES
 *
 * SYNOPSIS
 *    main()

 *
 * DESCRIPTION
 *     Utility to maintain message files used by form processor
 */

#include <stdtyp.h>
#include <stdio.h>
#include <fpparm.h>
#include <fpd.h>
#include <fpcode.h>
#include <fpemsg.h>
#include <ntm.h>

#define FILE_FORMAT  "iissmlib:msg%3.3s.msg"
```

```
#define MSG_MAX  100
#define MSG_PAGE  10

#define ENTER      0
#define NEXTPAGE   5
#define PREVPAGE   6
#define FIRSTPAGE  7
#define LASTPAGE   8

#define CONNECT_ERR "5"
#define PROGRAM_ERR "4"
#define NORMAL_TERM "1"

char  myname[] = "SDMMZZZZZZ";

#ifndef STANDALONE
int nmsgfil = 0;
struct msgfil msgfil[NMSGFIL];
#endif

static int    current = CURRNT;
static int    previous = PREV;
static MSGLIN msgs[MSG_MAX];
static FILE   *mmfp = NULL;

static bool saveit();
static void newfile();
```

```
main()
   {
   char     rcode[RCODE_LEN];
   char     base[5];
   char     prev_base[5];
   char     filedesc[30];
   MSGLIN   msg2[MSG_PAGE];
   int      page;
   int      pfkey;
   bool     changed = FALSE;
   int      curlin = 0;
   char     buffer[4096];
   char     *bufsize = "4096";
   char     systate;
   char     *trmstatus;
   static char fname[] = "MM        ";

   inital(buffer, bufsize, &systate, rcode);
   if (memcmp(rcode, NTMGOODRET, 5) != 0) trmnat(CONNECT_ERR);
   trmstatus = PROGRAM_ERR;
   initfp();
   opnfrm(fname, rcode);
   if (memcmp(rcode, OK, RCODE_LEN) != 0) goto error;
   addfrm(SCREN, fname, &page, rcode);
   if (memcmp(rcode, OK, RCODE_LEN) != 0) goto error;
   prev_base[0] = '\0';   /* guaranteed not to
                             match what's on screen */
   memset(msgs, ' ', sizeof msg2);
   for (;;)
      {
      oiscr(SCREN, &pfkey, rcode);
      if (memcmp(rcode, OK, RCODE_LEN) != 0) goto error;
      if (pfkey == QUITKEY) break;
      gdata(&current, "SCREEN.MM.MBASE", base, rcode);
      if (memcmp(rcode, OK, RCODE_LEN) != 0) goto error;
      memcpy(msg2, msgs+curlin, sizeof msg2);
      gdata(&current, "SCREEN.MM.MSGLIN", msgs+curlin, rcode);
      if (memcmp(rcode, OK, RCODE_LEN) != 0) goto error;
      changed = changed  ||  memcmp(msgs+curlin, msg2,
            sizeof msg2) != 0;
      if (memcmp(base, prev_base, sizeof base) != 0)
            /* new base entered */
         {
         if (changed) pmsgls("Changes NOT saved");
         newfile(filedesc, base);
         curlin = 0;
         changed = FALSE;
```

```
        memcpy(prev_base, base, sizeof prev_base);
         continue;
         }
      switch (pfkey)
         {
         case ENTER:
            if (changed && saveit(filedesc))
               {
               pmsgls("Changes saved");
               changed = FALSE;
               }
            else if (changed) pmsgls("Changes NOT saved");
            else pmsgls("No changes made");
            break;
         case NEXTPAGE:
            if (curlin + MSG_PAGE < MSG_MAX) curlin += MSG_PAGE;
            goto redisplay;
         case PREVPAGE:
            if (curlin >= MSG_PAGE) curlin -= MSG_PAGE;
            goto redisplay;
         case FIRSTPAGE:
            curlin = 0;
            goto redisplay;
         case LASTPAGE:
            while (curlin + MSG_PAGE < MSG_MAX  &&
                  memcmp(msgs[curlin + MSG_PAGE-1].name,
                  "        ", 8) != 0)
               curlin += MSG_PAGE;
         redisplay:
            pdata("SCREEN.MSGLIN", msgs + curlin, rcode);
            if (memcmp(rcode, OK, RCODE_LEN) != 0) goto error;
            break;
         }
      }
   trmstatus = NORMAL_TERM;
error:
   termfp();
   trmnat(trmstatus);
   }
bool saveit(filedesc)
   char filedesc[];
   {
   register int n;
   FILE *mmfp2;
```

```
    if (!(mmfp2 = fopen(filedesc, "w"))) return FALSE;
    for (n = MSG_MAX; n > 0  &&  memcmp(msgs[n - 1].name,
            "        ", 8) == 0;
          n--)
       ;
    n -= fwrite(msgs, sizeof (MSGLIN), n, mmfp2);
    return (fclose(mmfp2) == 0  &&  n == 0);
    }
static void newfile(filedesc, base)
    char filedesc[], base[];
    {
    register int num, i;
    char rcode[RCODE_LEN], charnum[6];

    memset(msgs, ' ', sizeof msgs);
    for (i = 0, num = 0; i < 3; i++)
       num = num*10 + base[i]-'0';
    num *= 100;
    sprintf(filedesc, FILE_FORMAT, base);
    if ((mmfp = fopen(filedesc, "r")) != NULL)
       {
       fread(msgs, sizeof (MSGLIN), MSG_MAX, mmfp);
       fclose(mmfp);
       }
    for (i = 0; i < MSG_MAX; i++)
       {
       sprintf(charnum, "%5d", num+i);
       memcpy(msgs[i].number, charnum, sizeof msgs[0].number);
       }
    pdata("SCREEN.MM.MSGLIN", msgs, rcode);
    }
```

SECTION 8

Layout Optimization System

8.1 Description

The Layout Optimization System provides the user with an easy method of designing and creating charts, either to the user's specifications or to default specifications using objects and relationships defined by the user.

8.1.1 Getting Started

In order to help the user define and print charts, the Layout System needs very specific inputs from the user. It also uses some specialized terminology to refer to these inputs.

An archetype is a template, or prototype. A chart archetype is a generic structure that you may want several of your specific charts to follow. An object archetype is a generic template which you may use for several different objects.

An icon is an instance of a symbol used on a chart.

A chart instance is a specific chart. An object instance is a specific chart. Using this terminology, a chart archetype might have several chart instances which use it, or an object archetype might have several object instances which use it.

Consider the following example chart:

Figure 8-1 Sample Chart

In order to use the Layout System to draw and/or change this chart, you need to create a form for each different type of symbol appearing on the chart. This process will create a template, or "archetype" for the symbol. The Layout system will use that form to create the symbol each time it is needed on the chart.

The above chart contains five different types of symbols. There are two different object types appearing on the chart (one with keys and one without), two different termination symbols (* and * P), and one categorization symbol. (Termination symbols and categorization symbols are not required by the Layout System. They are optional, to use only if your particular chart needs them.)

The first step towards using the Layout System is to create a form for each different type of symbol your chart needs. On each of those forms, you are going to "draw" one of the symbols using the "stretchy lines" provided by the User Interface (the syntax for using "stretchy lines" is found in the Form Editor User's Manual). You do not place text in the symbol. You use an item field on the form to reserve space for the text. The Layout System will take care of entering the text for you. For example, the form to define the type of symbol used by the "Offering" and "Section" objects might look like this:

```
+-------------------------------------------------------------+
|  +--------+ +--+                                            |
|  +--------+ +--+                                            |
|  +--------------+                                           |
|  |+------------+|                                           |
|  |+------------+|                                           |
|  |+------------+|                                           |
|  |+------------+|                                           |
|  |+------------+|                                           |
|  |+------------+|                                           |
|  |              |                                           |
|  +--------------+                                           |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
+-------------------------------------------------------------+
```

Figure 8-2 Sample Object Archetype Definition Form

The first two blocks on the form are the item fields into which the name of the object and the object number will be placed.

The large block is drawn, either using the "stretchy lines" provided by the User Interface (syntax is defined in the Form Editor User's Manual), or by using text characters (dashes, vertical bars, underlines, etc.). It is not recommended that text characters be used. The Layout System may need to resize some of the symbols you have defined when printing the chart. Stretchy lines can be resized, text characters cannot. This may result in gaps or overruns in a chart which uses text characters rather than stretchy lines.

The three smaller blocks within the large one are also item fields, into which the names of data fields required by the object are placed by the Layout System.

The two termination symbols used on the example chart are "*" and "* P". A separate form must be created for each of these symbols. This is done in the same manner as the object symbols are defined. An example might look like this:

```
+-------------------------------------------------------------+
| * P                                                         |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
|                                                             |
+-------------------------------------------------------------+
```

Figure 8-3 Sample Termination Symbol Archetype Definition Form

Notice that the "* P" symbol appears after the relationship "is implemented by", but after the "teaches" relationship it appears as "*
P". The form illustrated above can be used to represent both of these. The Layout System can rotate symbols so that they can fit into any context.

In the same way as the object and termination symbols were defined, the categorization symbol must also be defined on a form. Once every different type of symbol you plan to use on your chart has been defined on a form, you are ready to start using the Layout System.

There are two major pieces which make up the Layout Optimization System. There is a library of callable routines which may be accessed from within the user's application program. There is also an interactive program which the user can make use of to define a knowledge base of archetypes which the callable routines can use in creating charts. The knowledge base consists of 3 files, one for chart archetypes,
one for object archetypes, and one for relation archetypes. These files are created on the user's current directory.

8.2 Interactive Chart Definition

The interactive chart definition program is used to create and maintain a knowledge base of information which then can be used to create and manipulate charts. This program is accessed by entering "LAYOUT" in the function field on the IISS Function Screen. Figure 8-4 is the first screen to appear when the program is invoked:

```
                    Layout Optimization System
                              Main Menu
   Charts <PF5>            Objects <PF6>            Relations <PF7>

                                                     applcation
```

Figure 8-4

8.2.1 Chart Functions

When starting from scratch, the first thing you must do is define your chart. To start this process, press <PF5>. The following screen will appear:

```
                    Layout Optimization System
                         Chart Functions
Retreive Specific Item   <PF5>     Add      <PF9>     Delete    <PF16>
Retreive All             <PF8>     Update   <PF12>

MSG  C                                                  application
```

Figure 8-5

8.2.1.1 ADD Chart

To create a new chart, press <PF9> to ADD a chart. The following screen will appear:

```
                    Layout Optimization System
                         Chart Functions
Retreive Specific Item   <PF5>    Add      <PF9>    Delete    <PF16>
Retreive All             <PF8>    Update   <PF12>

                    Chart Name:

        Chart Orientation          Initial Layout Preference

            Vertical                   Left Downward
            Horizontal                 Left Upward

        Minimum Spacing Between Objects:   characters.

MSG:                                                   application
```

Figure 8-6

Valid entries are:

CHART NAME Up to 10 alpha-numeric characters

CHART ORIENTATION Place any non-blank character next to either vertical or horizontal to choose the orientation for your chart. ("Orientation" means the general way in which the objects will be positioned on the chart. A Vertical orientation means the chart will start at a specified point and progress either up or down. Horizontal means that the chart will begin at a specified point and progress to the right. If left blank, based on all other information the user enters, the software will choose the Chart Orientation at print time in order to optimize readability.

LAYOUT PREFERENCE Place any non-blank character next to your choice for initial layout preference. If you choose Left Downward, the chart will start on the left and progress to the right and down. If you choose Left Upward, the chart will start on the left and progress to the right and up. If left blank, based on all other information the user enters, the software will choose the Layout Preference at print time in order to optimize readability.

MINIMUM SPACING Enter the minimum number of spaces you wish to see between objects on your chart. This must be a numeric entry between 1 and 15. This field is optional. If left blank, the default is 1.

Once you have filled in the screen with your choices, press <ENTER>. The following screen will appear:

```
                    Layout Optimization System
                         Chart Functions
Retrieve Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retrieve All             <PF8>    Update   <PF12>

                    Chart Name:

        Chart Orientation          Initial Layout Preference

            Vertical                   Left Downward
            Horizontal                 Left Upward

        Minimum Spacing Between Objects:   characters.

MSG: 1 Item added  Enter data to add another or use QUIT          application
```

Figure 8-7

If you do not wish to create another chart, press <QUIT>. The screen illustrated in Figure 8-5 will return.

8.2.1.2 UPDATE Chart

If you wish to update a chart definition, while on the chart function main screen (Figure 8-5), press <PF12>. The screen shown in Figure 8-6 will appear. Type in the name of the chart you wish to change and the new data selections. (Any field left blank will default to its previous value.) Press <ENTER>. If the update was successful, the following will appear:

```
                    Layout Optimization System
                         Chart Functions
Retreive Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retreive All             <PF8>    Update   <PF12>

                    Chart Name:

        Chart Orientation          Initial Layout Preference

            [ ] Vertical                 [ ] Left Downward
            [ ] Horizontal               [ ] Left Upward

        Minimum Spacing Between Objects: [ ] characters.

MSG: [ ] Update successful.                          application
```

Figure 8-8

Press <QUIT> when finished updating.

8.2.1.3 DELETE Chart

To delete a chart, first press <PF16> from the Chart Functions Main Screen (Figure 8-5). The following screen will appear:

```
                    Layout Optimization System
                         Chart Functions
Retreive Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retreive All             <PF8>    Update   <PF12>

                    Chart Name:

MSG                                                   application
```

Figure 8-9

Type in the name of the chart you wish to delete. Press <ENTER>. The following will appear:

```
                    Layout Optimization System
                         Chart Functions
Retreive Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retrei·e All             <PF8>    Update   <PF12>

                    Chart Name: NEWCHART

MSG: 1 Successfully deleted.                          application
```

Figure 8-10

The chart and all of its related objects and relationships will be deleted.

Press <QUIT> to return to the Chart Functions Main Screen.

8.2.1.4 RETRIEVE Items

The last two options available under Chart Functions are Retrieve Specific Item and Retrieve All.

If you press <PF8> to retrieve all items, a list of the charts you have defined with their default values will appear.

```
                    Layout Optimization System
                         Chart Functions
Retrieve Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retrieve All             <PF8>    Update   <PF12>

                       Chart Descriptions

                 Chart                          Min
                 Name        Orient   Layout    Space
                HTEST          V        D        1
                ITEST          V        D        1
                JUNK           V        D        7
                MODULCHART     V        D        1
                NEWCHART       V        D        5
                XTEST          H        D       02

MSG  C                                                   application
```

Figure 8-11

You can use <TAB> to move your cursor to any one of the items. If you press <ENTER> with the cursor on one of the items, that item will become the default item. That means that any of the future choices you make will refer to that item until you change the default item. If you have not chosen a default item, the software will assume that the choices refer to the last item you added or updated.

If you press <PF5> to retrieve a specific item, the default item will appear on the screen (either the item you chose as default or the last item you added or deleted).

```
                    Layout Optimization System
                         Chart Functions
Retrieve Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retrieve All             <PF8>    Update   <PF12>

                    Chart Name: NEWCHART

        Chart Orientation          Initial Layout Preference

            [x] Vertical                [ ] Left Downward
            [ ] Horizontal              [x] Left Upward

        Minimum Spacing Between Objects: [5] characters.

MSG: [0]                                              application
```

Figure 8-12

From either of these options, press <QUIT> to return to the Chart Functions main screen.

Press <QUIT> again to return to the main menu (Figure 8-4).

8.2.2 Object Functions

Once your chart is defined, you need to define the objects contained on that chart. To choose the object functions, press <PF6> on the main menu. The following screen will appear:

```
                    Layout Optimization System
                         Object Functions
Retreive Specific Item    <PF5>     Add       <PF9>     Delite    <PF16>
Retreive All              <PF8>     Update    <PF12>

MSG                                                     application
```

Figure 8-13

8.2.2.1 ADD Object

If no objects exist, the first thing you must do is create one or more. Press <PF9> to ADD an object. The following screen will appear:

```
                    Layout Optimization System
                         Object Functions
Retreive Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retreive All             <PF8>    Update   <PF12>

                       Object Definition
                       For Chart:

        Object Name:                    Icon Name:

                      Connector Placement
            Originating                    Terminating
        [ ] Any Location               [ ] Any Location
        [ ] Bottom                     [ ] Bottom
        [ ] Top                        [ ] Top
        [ ] Right                      [ ] Right
        [ ] Left                       [ ] Left
        [ ] Bottom Right Corner        [ ] Bottom Right Corner
        [ ] Bottom Left Corner         [ ] Bottom Left Corner
        [ ] Top Right Corner           [ ] Top Right Corner
        [ ] Top Left Corner            [ ] Top Left Corner

MSG: [1] Item added. Enter data to add another or use QUIT          application
```

Figure 8-14

The following are valid entries:

CHART	Type the name of the chart the objects belong to
OBJECT NAME	Type the name of the object being defined
ICON NAME	Type the name of the icon containing the object
ORIGINATING CONNECTOR PLACEMENT	Place any non-blank character in the field next to your choice for the position on the object from which the connector is to originate (ex. "BOTTOM" means that you want the connector to originate from the bottom of the object you are currently defining).

TERMINATING
CONNECTOR PLACEMENT

Place any non-blank character in the field next to your choice for the position on the object to which the object you are defining is connected (ex. "TOP" means that you want the connector originating from the object you are currently defining to terminate into the top of the next object)

When the information is typed in as you want it, press <ENTER>. The following will appear:

Figure 8-15

If you wish to add more objects, continue entering data. Press <ENTER> after each set of data. To return to the Object Functions main screen (Figure 8-13), press <QUIT>.

8.2.2.2 UPDATE Object

To update an already existing object, press <PF12> from the Object Functions main screen (Figure 8-13). The screen shown in Figure 8-14 will appear containing the default object (if no default has been specified, it will contain the last object you added or updated). Change the data to the new settings (any field not changed will default to the previous entry; if there was no value previously, the software will automatically select settings based on all other information you enter) and press <ENTER>. The following will appear:

Figure 8-16

Press <QUIT> to return to the Object Functions main screen (Figure 8-13).

8.2.2.3 DELETE Object

To delete an object you have created, press <PF16> at the Object Functions main screen (Figure 8-13). The following will appear:

Figure 8-17

Enter the name of the chart and the name of the object to be deleted and press <ENTER>. The following will appear:

```
                    Layout Optimization System
                         Object Functions
Retrieve Specific Item   <PF5>     Add      <PF9>     Delete    <PF16>
Retrieve All             <PF8>     Update   <PF12>

                    Chart Name:   NEWCHART
                    Object Name:  OBJECT2

MSG. 1 Successfully deleted.                                  application
```

Figure 8-18

Press <QUIT> to return to the Object Functions main screen (Figure 8-13).

8.2.2.4 RETRIEVE Items

The last two options available under Object Functions are Retrieve Specific Item and Retrieve All.

If you press <PF8> to retrieve all items, the following screen will appear:

```
                    Layout Optimization System
                         Object Functions
Retrieve Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retrieve All             <PF8>    Update   <PF12>

                    Chart Name:
        Only objects for the specified chart will be retrieved.

MSG: 0                                                   application
```

Figure 8-19

Enter the name of the chart whose objects you want listed and press <ENTER>. A List of the objects you have defined for that chart with their default values will appear.

```
                    Layout Optimization System
                         Object Functions
Retrieve Specific Item   <PF5>     Add      <PF9>     Delete    <PF16>
Retrieve All             <PF8>     Update   <PF12>

                      Object Descriptions
                     For Chart: NEWCHART

        Object          Icon          Orig       Term
        Name            Name          Conn       Conn
        OBJECT1         ICON1          BOT        TOP
        OBJECT2         ICON2          BOT        TOP
        OBJECT3         ICON3          ANT        ANT

MSG: 0:                                                  application
```

Figure 8-20

You can use <TAB> to move your cursor to any one of the items. If you press <ENTER> with the cursor on one of the items, that item will become the default item. That means that any of the future choices you make will refer to that item until you change the default item. If you have not chosen a default item, the software will assume that the choices refer to the last item you added or updated.

If you press <PF5> to retrieve a specific item, the default item will appear on the screen (either the item you chose as default or the last item you added or deleted).

Figure 8-21

From either of these options, press <QUIT> to return to the Object Functions main screen.

8.2.3 Relation Functions

Now that the chart and the objects have been defined, you must define the relationships to be used in creating the chart. To do this you use the relation functions. At the Main Menu (Figure 8-4) press <PF7>. The following will appear:

```
                    Layout Optimization System
                       Relation Functions
Retrieve Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retrieve All             <PF8>    Update   <PF12>

MSG: 0                                                  application
```

Figure 8-22

8.2.3.1 ADD Relation

The following are valid entries for the ADD relation screen:

CHART	The name of the chart to which the relation applies.
RELATION	The name of the relation being created.
ORIGINATION ICON NAME	The name of the form which contains the template for the origination symbol to be used with this relation (if any).
TERMINATION ICON NAME	The name of the form which contains the template for the termination symbol to be used with this relation (if any).
COMPLEX ICON NAME	The name of the form which contains the template for the complex symbol to be used with this relation (if any).
ORIGINATING CONNECTOR PLACEMENT	Place any non-blank character in the field next to the location on the parent object from which you want the relation to start. If you do not choose, the Layout System will choose for you.
TERMINATING CONNECTOR PLACEMENT	Place any non-blank character in the field next to the location on the child object into which you want the relation to terminate. If you do not choose, the Layout System will choose for you.
SPECIAL CHARACTERISTICS TWO BEND LINE SLOPE	Place any non-blank character in the field next to the direction in which you want any sloped lines to progress. If you do not choose, the Layout System will choose for you.

SPECIAL CHARACTERISTICS

COMBINE RELATIONS — Place any non-blank character in the field next to "yes" or "no" depending upon whether you want like relations combined on your chart.

LINE STYLE — Enter the keyword for the type of line you wish to use to represent the relation (SOLID, DASH, or DOT), or you may leave blank. If you do not choose, the Layout System will choose for you.

When you have entered your choices, press <ENTER>. The following will appear:

Figure 8-23

If you wish to add more relations, enter the data and press <ENTER>.

When you have finished adding relations, press <*QUIT*> to return to the Relation Function screen (Figure 8-22).

8.2.3.2 UPDATE Relation

To update an existing relation, press <PF12>. The screen following will appear with either the relation you chose as default or the last relation you added or updated showing:

Figure 8-24

Make whatever changes you wish to the data and press <ENTER>. The following will appear:

```
                    Layout Optimization System
                       Relation Functions
Retrieve Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retrieve All             <PF8>    Update   <PF12>

                       Relation Definition
      For Chart:                  Relation:

                         Icon Names
   Origination:        Termination:          Complex:

     Connector Placement                Special Characteristics
Originating      Terminating          Two Bend        Combine
   Any Location     Any Location      Line Slope      Relations
   Bottom           Bottom              Up              Yes
   Top              Top                 Down            No
   Right            Right               Left
   Left             Left                Right

                Line Style:
MSG: 1 Update successful.                                  application
```

Figure 8-25

Press <QUIT> to return to the Relation Functions screen (Figure 8-22).

8.2.3.3 DELETE Relation

To delete an existing relation, press <PF16> at the Relation Functions screen. The following will appear:

```
                    Layout Optimization System
                       Relation Functions
Retrieve Specific Item   <PF5>    Add      <PF9>    Delete   <PF16>
Retrieve All             <PF8>    Update   <PF12>

                    Chart Name:    NEWCHART
                 Relation Name:    RELATION1

MSG: 0                                                  application
```

Figure 8-26

Enter the name of the chart to which the relation applies and the name of the relation you wish to delete. Press <ENTER>. The following will appear:

```
                    Layout Optimization System
                       Relation Functions
Retrieve Specific Item    <PF5>      Add       <PF9>      Delete    <PF16>
Retrieve All              <PF8>      Update    <PF12>

                    Chart Name:    NEWCHART
                 Relation Name:    RELATION1

MSG: 1 Successfully deleted.                                  application
```

Figure 8-27

Press <QUIT> to return to the Relation Functions screen (Figure 8-22).

8.2.3.4 RETRIEVE Items

The last two options available under Relation Functions are Retrieve Specific Item and Retrieve All.

If you press <PF8> to retrieve all items, the following screen will appear:

```
                    Layout Optimization System
                       Relation Functions
Retrieve Specific Item   <PF5>     Add      <PF9>     Delete    <PF16>
Retrieve All             <PF8>     Update   <PF12>

                    Chart Name:
        Only relations for the specified chart will be retrieved.

MSG: 0                                                  application
```

Figure 8-28

Enter the name of the chart whose relations you want listed and press <ENTER>. A list of the relations you have defined for that chart with their default values will appear.

```
                    Layout Optimization System
                       Relation Functions
Retrieve Specific Item    <PF5>      Add       <PF9>      Delete    <PF16>
Retrieve All              <PF8>      Update    <PF12>

                    Relation Descriptions
                    For Chart: NEWCHART

                                                          Two
                                                   Orig   Term  Bend  Comb
Relation    Origin     Term     Complex   Line Style Conn   Conn  Line  Bel
RELATION1              ICON2    ICON3      SOLID     B      T     U     T
RELATION2  ICON1       ICON2               SOLID     B      T     U     T
RELATION3              ICON2               SOLID     B      T     U     T

MSG: 0                                                          application
```

Figure 8-29

You can use <TAB> to move your cursor to any one of the items. If you press <ENTER> with the cursor on one of the items, that item will become the default item. That means that any of the future choices you make will refer to that item until you change the default item. If you have not chosen a default item, the software will assume that the choices refer to the last item you added or updated.

If you press <PF5> to retrieve a specific item, the default item will appear on the screen (either the item you chose as default or the last item you added or deleted).

Figure 8-30

From either of these options, press <QUIT> to return to the Relation Functions main screen.

8.3 Callable Routines

This section of the manual describes the callable routines used by the Layout Optimization System. They are listed alphabetically with a brief functional description in Table 8.1 below:

TABLE 8.1 Callable Layout System Routines

ROUTINE	FUNCTION
BEGCHT	Change the application program's display list to that of the chart identified
DCHART	Describe a chart archetype
DELCHT	Delete all object and relation definitions for a named chart instance
DELOBJ	Delete an object instance from a chart instance
DELREL	Delete a relationship instance from a chart instance
DOBJTP	Describe an object archetype
DRELTP	Describe a relation archetype
DRWCHT	Layout the objects and relationships for a chart
ENDCHT	Return to the application program's original display list
GINSNM	Generate an instance name
INITFL	Initialize the Layout Optimization System
MAKCHT	Create an instance of a chart archetype
MAKOBJ	Create an instance of an object archetype
MAKREL	Create an instance of a relation archetype
PRNCHT	Display a chart
TERMFL	Terminate the Layout Optimization System

The use of each routine is explained in detail in the following sections. This includes a description of the purpose of the routine, the format for calling the routine, descriptions of the calling parameters, and an example call. Note that the calling formats and example are given in COBOL and Str and Num in the parameter format mean character string and long integer respectively. Example data types in COBOL, PL/I, C, and Fortran are described in Table 8.2:

TABLE 8.2 Example Data Types

	COBOL	PL/I	C	Fortran
Num	S9(5) COMP	FIXED BIN(31)	int	INTEGER
Str	PIC X(n)	CHAR(N)	char[n]	CHARACTER*N

NOTE: A C character string may contain the null character if it is after the last character the FP looks at (e.g. after the semicolon in a qualified path name).

NOTE: PMSGLC is a Form Processor callable routine which appears in most of the examples included in this section. The description can be found in the Form Processor User's Manual.

NOTE: "OK" is a variable name defined in the include member FPCODE.

8.3.1 BEGCHT

This routine is used to switch the application program's display list to that of the specified chart. BEGCHT is required to enable the application program to have access to the forms used to pictorially represent the objects and relationships of the chart. Once the application has access to the forms containing the archetypes, it can place the correct names, etc. into the pictures to make them specific to each instance.

8.3.1.1 Calling Format

```
CALL "BEGCHT" USING CHART-INSTANCE-NAME,
                    RCODE.
```

8.3.1.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
CHART-INSTANCE-NAME	Input	Str(10)	The name of the chart.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.1.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01  CHART-INSTANCE-NAME          PIC X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "BEGCHT" USING CHART-INSTANCE-NAME
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.2 DCHART

This routine is used to describe a new chart archetype, or generic prototype. The user specifies the general guidelines he or she desires the Layout System to use when designing the chart. These preferences are specified by including a list of keywords. The user must specify the length, in characters, of the the keyword list (e.g. if the user specifies the list vertical,leftup,minspace=2 the keyword list length is 26).

8.3.2.1 Calling Format

```
CALL "DCHART" USING CHART-ARCHETYPE-REFERENCE-NAME,
                    KEYWORD-LIST-LENGTH,
                    KEYWORD-LIST,
                   RCODE.
```

8.3.2.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
CHART-ARCHETYPE-REFERENCE-NAME	Input	Str(10)	The reference name for the chart archetype.
KEYWORD-LIST-LENGTH	Input	Num	Specifies the length (in characters) of the keyword list.
KEYWORD-LIST	Input	Str	A list of chart characteristic keywords separated by commas. Refer to Section 8.3.2.3 below for valid keywords.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.2.3 Valid Keywords

```
VERTICAL     - provide vertical object orientation
HORIZONTAL   - provide horizontal object orientation
LEFTDOWN     - provide left downward initial layout
               structure
LEFTUP       - provide left upward initial layout
               structure
MINSPACE=99  - minimum spacing between objects
```

8.3.2.4 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01 CHART-ARCHETYPE-REFERENCE-NAME PIC  X(10).
01 KEYWORD-LIST-LENGTH            PIC S9(05) COMP VALUE 30.
01 KEYWORD-LIST                   PIC  X(30).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "DCHART" USING CHART-ARCHETYPE-REFERENCE-NAME,
                         KEYWORD-LIST-LENGTH,
                         KEYWORD-LIST,
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.3 DELCHT

This routine deletes all object and relation definitions and frees all computer resources which were acquired for the named chart instance.

8.3.3.1 Calling Format

```
CALL "DELCHT" USING CHART-INSTANCE-NAME,
                    RCODE.
```

8.3.3.2 Parameter Descriptions

NAME	I/O	FORMAT	DEFINITION
CHART-INSTANCE-NAME	Input	Str(10)	The name of the chart for which the object and relation definitions will be deleted.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.3.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01  CHART-INSTANCE-NAME     PIC  X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "DELCHT" USING CHART-INSTANCE-NAME,
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.4 DELOBJ

This routine deletes the named object instance from a chart instance.

8.3.4.1 Calling Format

```
CALL "DELOBJ" USING CHART-INSTANCE-NAME,
                    OBJECT-INSTANCE-NAME,
                    RCODE.
```

8.3.4.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
CHART-INSTANCE-NAME	Input	Str(10)	The instance name of the chart containing the object to be deleted.
OBJECT-INSTANCE-NAME	Input	Str(10)	The instance name of the object to be deleted.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.4.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01  CHART-INSTANCE-NAME      PIC  X(10).
01  OBJECT-INSTANCE-NAME     PIC  X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "DELOBJ" USING CHART-INSTANCE-NAME,
                         OBJECT-INSTANCE-NAME,
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.5 DELREL

This routine is used to delete a relationship instance from a chart.

8.3.5.1 Calling Format

```
CALL "DELREL" USING CHART-INSTANCE-NAME,
                    RELATIONSHIP-INSTANCE-NAME,
                    PARENT-OBJECT-INSTANCE-NAME,
                    CHILD-OBJECT-INSTANCE-NAME,
                    RCODE.
```

8.3.5.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
CHART-INSTANCE-NAME	Input	Str(10)	The instance name of the chart containing the relationship to be deleted
RELATION-SHIP-INSTANCE-NAME	Input	Str(10)	The instance name of the relationship to be deleted
PARENT-OBJECT-INSTANCE-NAME	Input	Str(10)	The instance name of the parent or origination object of the relationship
CHILD-OBJECT-INSTANCE-NAME	Input	Str(10)	The instance name of the child or termination object of the relationship
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.5.3 Example

```
     WORKING-STORAGE SECTION.
     COPY FPCODE OF IISSCLIB.
     COPY FPPARM OF IISSCLIB.
     01  CHART-INSTANCE-NAME            PIC  X(10).
    01  RELATIONSHIP-INSTANCE-NAME    PIC  X(10).
     01  PARENT-OBJECT-INSTANCE-NAME   PIC  X(10).
     01  CHILD-OBJECT-INSTANCE-NAME    PIC  X(10).
     PROCEDURE DIVISION.
          .
          .
          .
          CALL "DELREL" USING CHART-INSTANCE-NAME,
                              RELATIONSHIP-INSTANCE-NAME,
                              PARENT-OBJECT-INSTANCE-NAME,
                              CHILD-OBJECT-INSTANCE-NAME,
                             RCODE.
          IF RCODE = OK
          .
          .
          .
          ELSE
             CALL "PMSGLC" USING RCODE.
```

8.3.6 DOBJTP

This routine is used to describe an object archetype. You must specify the name of the chart archetype which will use the object archetype being created, the name of the object archetype being created, the name of the form where the object archetype is defined, a keyword list containing keywords to specify connections should originate and terminate, and the length (in characters) of the keyword list. For example, if the keyword list org_bottom,trm_top is used, the keyword list length is 18. The keyword list is optional. If none is specified, the Layout System will choose the positioning of connectors.

8.3.6.1 Calling Format

```
CALL "DOBJTP" USING CHART-ARCHETYPE-REFERENCE-NAME,
                    OBJECT-ARCHETYPE-REFERENCE-NAME,
                    OBJECT-FORM-NAME,
                   KEYWORD-LIST-LENGTH,
                    KEYWORD-LIST,
                   RCODE.
```

8.3.6.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
CHART-ARCHETYPE-REFERENCE-NAME	Input	Str(10)	The name of the chart archetype to which the description being defined is to be assigned.
OBJECT-ARCHETYPE-REFERENCE-NAME	Input	Str(10)	The name of the object archetype to be defined.
OBJECT-FORM-NAME	Input	Str(10)	The name of the form to be used as the pictorial representation of this object.
KEYWORD-LIST-LENGTH	Input	Num	The length of the keyword list.
KEYWORD-LIST	Input	Str	A list of object characteristic keywords separated by commas. Refer to Section 8.3.6.3 below for valid keywords.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.6.3 Valid Keywords

ORG_BOTTOM - all originating connections should be located at the bottom of the object.
ORG_TOP - all originating connections should be located at the top of the object.
ORG_LEFT - all originating connections should be located to the left of the object.
ORG_RIGHT - all originating connections should be located to the right of the object.
TRM_BOTTOM - all terminating connections should be located at the bottom of the object.
TRM_TOP - all terminating connections should be located at the top of the object.

TRM_LEFT - all terminating connections should be located to the left of the object.
TRM_RIGHT - all terminating connections should be located to the right of the object.
ORG_LOW_R - all originating connections should be located on the bottom right-hand corner of the object.
ORG_LOW_L - all originating connections should be located on the bottom left-hand corner of the object.
ORG_UP_L - all originating connections should be located on the upper left-hand corner of the object.
ORG_UP-R - all originating connections should be located on the upper right-hand corner of the object.
TRM_LOW_R - all terminating connections should be located on the bottom right-hand corner of the object.
TRM_LOW_L - all terminating connections should be located on the bottom left-hand corner of the object.
TRM_UP_R - all terminating connections should be located on the upper right-hand corner of the object.
TRM_UP_L - all terminating connections should be located on the upper left-hand corner of the object.

8.3.6.4 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01 CHART-ARCHETYPE-REFERENCE-NAME  PIC  X(10).
01 OBJECT-ARCHETYPE-REFERENCE-NAME PIC  X(10).
01 OBJECT-FORM-NAME                PIC  X(10).
01 KEYWORD-LIST-LENGTH             PIC S9(5) COMP VALUE 30.
01 KEYWORD-LIST                    PIC  X(30).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "DOBJTP" USING CHART-ARCHETYPE-REFERENCE-NAME,
                         OBJECT-ARCHETYPE-REFERENCE-NAME,
                         OBJECT-FORM-NAME,
                        KEYWORD-LIST-LENGTH,
                         KEYWORD-LIST,
                        RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.7 DRELTP

This routine is used to describe a relation archetype. You must specify the name of the chart archetype to which the relation applies, the name of the relation being created, the name of the forms containing the definitions of the termination and origination symbols, the name of the form containing the complex symbol used (complex symbol is any symbol which connects one parent object to two or more child objects), the style of the line used to represent the relation, a list of keywords to describe the relationship, and the length of the keyword list in characters. For example, if the keyword list org_right,trm_left,slopedown is used the keyword list length is 28.

8.3.7.1 Calling Format

```
CALL "DRELTP" USING CHART-ARCHETYPE-REFERENCE-NAME,
                    RELATION-ARCHETYPE-REFERENCE-NAME,
                    TERMINATION-SYMBOL-FORM-NAME,
                    ORIGINATION-SYMBOL-FORM-NAME,
                    COMPLEX-SYMBOL-FORM-NAME,
                    LINE-STYLE,
                    KEYWORD-LIST-LENGTH,
                    KEYWORD-LIST,
                    RCODE.
```

8.3.7.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
CHART-ARCHETYPE-REFERENCE-NAME	Input	Str(10)	The name of the chart archetype to which the description being defined is to be assigned.
RELATION-ARCHETYPE-REFERENCE-NAME	Input	Str(10)	The name of the relationship archetype to be defined.
TERMINATION-SYMBOL-FORM-NAME	Input	Str(10)	The name of the form to be used as the pictorial representation of the termination symbol of this relation.
ORIGINATION-SYMBOL-FORM-NAME	Input	Str(10)	The name of the form to be used as the pictorial representation of the origination symbol of this relation.
COMPLEX-SYMBOL-FORM-NAME	Input	Str(10)	The name of the form to be used as the pictorial representation of a complex relation.
LINE-STYLE	Input	Str(10)	The stretchy line style definition for the connector line expressed as a keyword. Refer to Section 8.3.7.3 below for valid keywords.
KEYWORD-LIST-LENGTH	Input	Num	Specifies the length (in characters) of the keyword list.
KEYWORD-LIST	Input	Str	A list of relation characteristic keywords separated by commas. Refer to Section 8.3.7.4 for valid keywords.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.7.3 Valid Keywords for Line-Style

SOLID - the line is to be represented by a solid line.
DASH - the line is to be represented by a dashed line.
DOT - the line is to be represented by a dotted line.
DEFAULT - the Form Processor will determine the style of the line.

8.3.7.4 Valid Keywords for Keyword-List

ORG_RIGHT - the relationship originates from the right-hand side of an object.
ORG_LEFT - the relationship originates from the left-hand side of an object.
ORG_BOTTOM - the relationship originates from the bottom of an object.
ORG_TOP - the relationship originates from the top of an object.
TRM_RIGHT - the relationship terminates on the right-hand side of an object.
TRM_LEFT - the relationship terminates on the left-hand side of an object.
TRM_BOTTOM - the relationship terminates on the bottom of an object.
TRM_TOP - the relationship terminates on the top of an object.
SLOPEUP - the relationship has two bends and slopes upward.
SLOPEDOWN - the relationship has two bends and slopes downward.
SLOPELEFT - the relationship has two bends and slopes to the left.
SLOPERIGHT - the relationship has two bends and slopes to the right.
COMBINE - pictorially combine relationships of this type, whenever possible, when the chart is drawn.

8.3.7.5 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01 CHART-ARCHETYPE-REFERENCE-NAME     PIC  X(10).
01 RELATION-ARCHETYPE-REFERENCE-NAME  PIC  X(10).
01 TERMINATION-SYMBOL-FORM-NAME       PIC  X(10).
01 ORIGINATION-SYMBOL-FORM-NAME       PIC  X(10).
01 COMPLEX-SYMBOL-FORM-NAME           PIC  X(10).
01 LINE-STYLE                         PIC  X(10).
01 KEYWORD-LIST-LENGTH                PIC S9(5) COMP
                                               VALUE 30.
01 KEYWORD-LIST                       PIC  X(30).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "DRELTP" USING CHART-ARCHETYPE-REFERENCE-NAME,
                         RELATION-ARCHETYPE-REFERENCE-NAME,
                         TERMINATION-SYMBOL-FORM-NAME,
                         ORIGINATION-SYMBOL-FORM-NAME,
                         COMPLEX-SYMBOL-FORM-NAME,
                         LINE-STYLE,
                         KEYWORD-LIST-LENGTH,
                         KEYWORD-LIST,
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.8 DRWCHT

This routine is used to layout the objects and relationships for a chart.

8.3.8.1 Calling Format

```
CALL "DRWCHT" USING CHART-INSTANCE-NAME,
                    RCODE.
```

8 3.8.2 Parameter Descriptions

NAME	I/O	FORMAT	DESCRIPTION
CHART-INSTANCE-NAME	Input	Str(10)	The instance name of the chart.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.8.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01 CHART-INSTANCE-NAME                PIC  X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "DRWCHT" USING CHART-INSTANCE-NAME,
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.9 ENDCHT

This routine returns the application program's display list from the chart specified in the BEGCHT routine to the application program's original display list.

8.3.9.1 Calling Format

```
CALL "ENDCHT" USING RCODE.
```

8.3.9.2 Parameter Descriptions

NAME	I/O	Format	Definition
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.9.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "ENDCHT" USING RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.10 GINSNM

This routine is used to generate an instance name. It is provided to give the application program a means to create a unique instance name which can be used in the MAKCHT, MAKOBJ or MAKREL routine calls. Use of this routine is not required.

8.3.10.1 Calling Format

```
CALL "GINSNM" USING INSTANCE-NAME,
                    RCODE.
```

8.3.10.2 Parameter Descriptions

NAME	I/O	Format	Definition
INSTANCE-NAME	Output	Str(10)	The generated instance name. This name will be nine characters long and it will be unique.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.10.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01  INSTANCE-NAME     PIC  X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "GINSNM" USING INSTANCE-NAME,
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.11 INITFL

This routine is used to acquire and initialize the computer resources that will be required by the application program.

8.3.11.1 Calling Format

CALL "INITFL" USING RCODE.

8.3.11.2 Parameter Description

NAME	I/O	Format	Definition
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member PFCODE.

8.3.11.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "INITFL" USING RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.12 MAKCHT

This routine is used to create an instance of a chart from the chart archetype reference name specified. The archetype can be one created at run-time, or it can be one which was previously created and resides in a knowledge base.

8.3.12.1 Calling Format

```
CALL "MAKCHT" USING CHART-INSTANCE-NAME,
                    CHART-ARCHETYPE-REFERENCE-NAME,
                    RCODE.
```

8.3.12.2 Parameter Descriptions

NAME	I/O	Format	Definition
CHART-INSTANCE-NAME	Input	Str(10)	The instance name of the chart to be created.
CHART-ARCHETYPE-REFERENCE-NAME	Input	Str(10)	The reference name of the chart archetype to use to create the chart.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.12.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01  CHART-INSTANCE-NAME                    PIC  X(10).
01  CHART-ARCHETYPE-REFERENCE-NAME         PIC  X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "MAKCHT" USING CHART-INSTANCE-NAME,
                         CHART-ARCHETYPE-INSTANCE-NAME,
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.13 MAKOBJ

This routine is used to create an instance of the object archetype using the object archetype specified. The object archetype can be created at run-time or it can be on which was previously created and which exists in the knowledge base.

8.3.13.1 Calling Format

```
CALL "MAKOBJ" USING CHART-INSTANCE-NAME,
                    OBJECT-INSTANCE-NAME,
                    OBJECT-ARCHETYPE-REFERENCE-NAME,
                    RCODE.
```

8.3.13.2 Parameter Descriptions

NAME	I/O	Format	Definition
CHART-INSTANCE-NAME	Input	Str(10)	The instance name of the chart to which the object being created is to be assigned.
OBJECT-INSTANCE-NAME	Input	Str(10)	The instance name of the object which is being created.
OBJECT-ARCHETYPE-REFERENCE-NAME	Input	Str(10)	The reference name of the object archetype to use to create the object.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.13.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01  CHART-INSTANCE-NAME                    PIC  X(10).
01  OBJECT-INSTANCE-NAME                   PIC  X(10).
01  OBJECT-ARCHETYPE-REFERENCE-NAME        PIC  X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "MAKOBJ" USING CHART-INSTANCE-NAME,
                         OBJECT-INSTANCE-NAME,
                         OBJECT-ARCHETYPE-INSTANCE-NAME,
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.14 MAKREL

This routine is used to create an instance of a relation archetype using the relation archetype specified. The relation archetype can be created at run-time or it can be one which was created previously and exists in the knowledge base.

8.3.14.1 Calling Format

```
CALL "MAKREL" USING CHART-INSTANCE-NAME,
                    RELATIONSHIP-INSTANCE-NAME,
                    PARENT-OBJECT-INSTANCE-NAME,
                    CHILD-OBJECT-INSTANCE-NAME,
                    RELATION-ARCHETYPE-REFERENCE-NAME,
                    LABEL-LENGTH,
                    LABEL,
                    RCODE.
```

8.3.14.2 Parameter Descriptions

NAME	I/O	Format	Definition
CHART-INSTANCE-NAME	Input	Str(10)	The instance name of the chart to which this relationship is to be assigned.
RELATION-SHIP-INSTANCE-NAME	Input	Str(10)	The instance name of the relation being created. This name has a has a maximum of 9 characters; the last character must be blank. This is so that origination, termination, and complex form instances can be referenced in the application program by adding "O", "T", or "C" to the end of the relationship instance name.
PARENT-OBJECT-INSTANCE-NAME	Input	Str(10)	The instance name of the parent (origination) object.
CHILD-OBJECT-INSTANCE-NAME	Input	Str(10)	The instance name of the child (termination) object.
RELATION-ARCHETYPE-REFERENCE-NAME	Input	Str(10)	The reference name of the relation archetype from which to create the relationship.
LABEL-LENGTH	Input	Num	The length of the text string in in the label parameter.
LABEL	Input	Str	A non-modifiable text string to be associated with the relation.
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member FPCODE.

8.3.14.3 Example

```
     WORKING-STORAGE SECTION.
     COPY FPCODE OF IISSCLIB.
     COPY FPPARM OF IISSCLIB.
     01  CHART-INSTANCE-NAME                        PIC  X(10).
    01  RELATIONSHIP-INSTANCE-NAME                 PIC  X(10).
     01  PARENT-OBJECT-INSTANCE-NAME                PIC  X(10).
     01  CHILD-OBJECT-INSTANCE-NAME                 PIC  X(10).
     01  RELATION-ARCHETYPE-REFERENCE-NAME          PIC  X(10).
     01  LABEL-LENGTH                               PIC S9(05) COMP.
     01  LABEL                                      PIC  X(??).
     PROCEDURE DIVISION.
          .
          .
          .
          CALL "MAKREL" USING CHART-INSTANCE-NAME,
                              RELATIONSHIP-INSTANCE-NAME,
                              PARENT-OBJECT-INSTANCE-NAME,
                              CHILD-OBJECT-INSTANCE-NAME,
                              RELATION-ARCHETYPE-REFERENCE-NAME,
                              LABEL-LENGTH,
                              LABEL,
                              RCODE.
          IF RCODE = OK
          .
          .
          .
          ELSE
             CALL "PMSGLC" USING RCODE.
```

8.3.15 PRNCHT

This routine is used to display a chart on the device specified.

8.3.15.1 Calling Format

```
CALL "PRNCHT" USING CHART-INSTANCE-NAME,
                    DISPLAY-DEVICE-TYPE,
                    DISPLAY-DEVICE-NAME,
                    PAGINATION-STYLE,
                    RCODE.
```

8.3.15.2 Parameter Descriptions

NAME	I/O	Format	Definition
CHART-INSTANCE-NAME	Input	Str(10)	The instance name of the chart to be displayed.
DISPLAY-DEVICE-TYPE	Input	Str(10)	The type of device to be used for the display (e.g. LN03).
DISPLAY-DEVICE-NAME	Input	Str(10)	The name of the device to be used for the display.
PAGINATION-STYLE	Input	Str(10)	A keyword used to specify the pagination style (see section 8.3.15.3 for a list of valid keywords).
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member PFCODE.

8.3.15.3 Pagination Style Keywords

LOGICAL - Provide logical pagination.
PHYSICAL - Provide physical pagination.
BROWSE - Allow the user a "read-only" means of examining forms. The user may enter window manager mode and scroll the form around. Application function keys are not enabled and data is not enterable.

8.3.15.4 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
01  CHART-INSTANCE-NAME                    PIC  X(10).
01  DISPLAY-DEVICE-TYPE                    PIC  X(10).
01  DISPLAY-DEVICE-NAME                    PIC  X(10).
01  PAGINATION-STYLE                       PIC  X(10).
PROCEDURE DIVISION.
     .
     .
     .
     CALL "PRNCHT" USING CHART-INSTANCE-NAME,
                         DISPLAY-DEVICE-TYPE,
                         DISPLAY-DEVICE-NAME,
                         PAGINATION-STYLE,
                         RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

8.3.16 TERMFL

This routine is used to relinquish the computer resources acquired during the use of the system.

8.3.16.1 Calling Format

```
CALL "TERMFL" USING RCODE.
```

8.3.16.2 Parameter Descriptions

NAME	I/O	Format	Definition
RCODE	Output	Str(5)	The routine return code. The possible values are defined in the include member PFCODE.

8.3.16.3 Example

```
WORKING-STORAGE SECTION.
COPY FPCODE OF IISSCLIB.
COPY FPPARM OF IISSCLIB.
PROCEDURE DIVISION.
     .
     .
     .
     CALL "TERMFL" USING RCODE.
     IF RCODE = OK
     .
     .
     .
     ELSE
        CALL "PMSGLC" USING RCODE.
```

SECTION 9

THREE DIMENSIONAL (3-D) GRAPHICS

UI-PLUS has the capability of providing three dimensional (3-D) wire-frame graphics. This ability has been implemented in accordance with the PHIGS (Programmer's Hierarchical Interactive Graphics System) standard, ANSI X3.144-198.

The basic philosophy of PHIGS is that a graphic picture is created by grouping together commands which create different parts of the picture. For example, an arrow may be drawn by grouping together the commands to draw 2 polylines. These groups of commands are called structures. A more complex picture may be drawn by grouping together commands to draw parts of the picture and commands which pull other structures into the main structure. For instance, a car may be drawn by using commands to draw the lines which represent the outlines of the body of the car, and commands to pull in a separate structure which draws a wheel. The wheel structure may be pulled in 4 times, once for each wheel. Commands may be included which reposition the wheel structure each time it is used, so that the same structure used for the left front wheel can be rotated and moved to the correct position for the right rear wheel, etc.

Once the structure to draw a picture is built, that structure can be displayed on a workstation. Usually, a workstation corresponds to a physical device (e.g., a terminal screen), but the UI-PLUS implementation requires the user to reference each UI window as a workstation. In this manner, several pictures can be drawn on a single screen at the same time.

More detailed information about PHIGS is contained in ANSI X3.144-198x, Computer Graphics - Programmer's Hierarchical Interactive Graphics System (PHIGS) Functional Description. This user's manual will explain the use of each of the individual callable routines currently in place for use with UI-PLUS.

The UI-PLUS implementation of PHIGS 3-D graphics consists entirely of routines which adhere to the PHIGS C binding. These PHIGS routines are not callable from Fortran or COBOL. (While UI-PLUS has not implemented every function described in the PHIGS standard, all functionality which has been implemented by UI-PLUS has been implemented according to the standard.) The PHIGS callable routines may be embedded into ADL code. This is done in exactly the same manner as C code is embedded in ADL, by enclosing the calls between the special characters %{ and %}. The reason that it is permissible to embed the routines into ADL is that the ADL will be translated into C code by the Application Generator.

The examples in this manual are written in C.

The appearance of an ellipsis (...) signifies that further statements may appear at this point but are not immediately necessary to the specific example.

Calls which are necessary to the specific example are shown in bold face type.

All calls MUST be made with the name of the routine in lower case.

9.1 POPENPHIGS

This routine is used to allocate and initialize the PHIGS state list. The error file specification and buffer space size are recorded, but not otherwise used in the UI-PLUS implementation. All PHIGS errors encountered in processing are written to the NTM error log. The program must call INITFP before opening PHIGS.

9.1.1 Calling Sequence

```
popenphigs(error_file,buf_size);
```

9.1.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
ERROR_FILE	INPUT	Pchar*	The address of the null-terminated string which specifies the name of the file into which the errors are to be logged (see footnote).
BUF_SIZE	INPUT	Plong	A 4-byte signed integer which specifies the size (in bytes) of the buffer space to be reserved for the error file.

FOOTNOTE: The UI-PLUS implementation of PHIGS automatically writes all PHIGS errors encountered into the NTM error log. The ERROR_FILE and BUF_SIZE information is recorded, but not otherwise used.

9.1.3 Example

```
Pchar error_file[] = "filename";
Plong buf_size = 512; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
```

9.2 PCLOSEPHIGS

PCLOSEPHIGS is used to free the PHIGS state list and any other dynamically allocated structures. All open workstations must be closed before closing PHIGS. In order to terminate the application, the program must call TERMFP.

9.2.1 Calling Sequence

```
pclosephigs();
```

9.2.2 Parameter Descriptions

PCLOSEPHIGS has no associated arguments.

9.2.3 Example

```
Pchar error_file[] = "filename";
Plong buf_size = 512; ...

initfp(); ...
popenphigs(error_file, buf_size); ...

pclosephigs(); ...
termfp()
```

9.3 POPENSTRUCT

POPENSTRUCT opens the specified structure so that elements (instructions) may be added. If the structure has not yet been created, POPENSTRUCT also creates the structure. Only one structure may be opened at a time. In order to open a new structure, the currently open structure (if any) must be closed.

9.3.1 Calling Sequence

```
popenstruct(struct_id);
```

9.3.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
STRUCT_ID	INPUT	Pint	A 4-byte signed integer which which serves as the identifier for the structure to be opened.

9.3.3 Example

```
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...

initfp(); ...
popenphigs(error_file, buf_size); ...

popenstruct(struct_id); ...

pclosephigs(); ...
termfp();
```

9.4 PCLOSESTRUCT

PCLOSESTRUCT closes the currently open structure. Only one structure may be open at a time. In order to open a new structure, the currently open structure (if any) must be closed.

9.4.1 Calling Sequence

```
pclosestruct();
```

9.4.2 Parameter Descriptions

PCLOSESTRUCT has no associated arguments.

9.4.3 Example

```
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

9.5 PEXECUTESTRUCT

PEXECUTESTRUCT inserts an execute structure element (an instruction which points to another structure) into the currently open structure. When the structure containing the execute structure element (the "parent" structure) is traversed, at the point where the execute structure element is reached, the traversal state list for the parent structure is saved, and the structure pointed to (the "child" structure) is completely traversed using the child structure's traversal state list. When the child structure traversal is complete, the original traversal state list is restored, and traversal of the parent structure is continued from the point where it left off.

For example, if in order to create a structure which draws a car, there may be several instructions to draw different parts of the car, and another structure which draws wheels. Included in the structure for the car would be instructions to draw the doors, hood, fenders, etc., and an execute structure element instruction which, in effect, says "execute the structure which draws a wheel". This execute structure element would appear four times in the car structure in order to draw four wheels.

9.5.1 Calling Sequence

```
pexecutestruct(struct_id);
```

9.5.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
STRUCT_ID	INPUT	Pint	A 4-byte signed integer which specifies the child structure to be traversed.

9.5.3 Example

```
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id1 = 1;
Pint struct_id2 = 2; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id1); ...
pexecutestruct(struct_id2); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

9.6 PDELSTRUCT

PDELSTRUCT unposts the specified structure from all workstations and deletes the specified structure and all references to it from memory. If the specified structure is currently open, it is closed prior to deletion and re-opened afterward.

9.6.1 Calling Sequence

```
pdelstruct(struct_id);
```

9.6.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
STRUCT_ID	INPUT	Pint	A 4-byte signed integer which specifies the structure to be deleted.

9.6.3 Example

```
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id1 = 1;
Pint struct_id2 = 2; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id1); ...
pexecutestruct(struct_id2); ...
pclosestruct(); ...
pdelstruct(struct_id1); ...
pclosephigs(); ...
termfp();
```

9.7 PDELSTRUCTNET

PDELSTRUCTNET unposts the specified structure together with all of its child structures (depending upon the reference flag), from all workstations, and deletes the specified structure and all references to it from memory. Any structure in the specified network that is currently open is closed.

9.7.1 Calling Sequence

```
pdelstructnet(struct_id, ref_flag);
```

9.7.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
STRUCT_ID	INPUT	Pint	A 4-byte signed integer which specifies the structure which is to be deleted.
REF_FLAG	INPUT	Preff	A keyword which specifies whether to delete child structures which are referenced by other parent structures. PDELETE - delete the parent structure and all of its child structures, even if the child structures are referenced by other structures. PKEEP - delete the parent structure and all of the child structures which are not referenced by other structures.

9.7.3 Example

Consider the following diagram:

```
                                +-  Structure 3
                                |   (Auto Body)        Structure 6
  Structure 1 (Automobile)    --+                  +-    (Tire)
                                |   Structure 5 --+
                                +-    (Wheel)     +-  Structure 7
                                                       (Hubcap)

                                                     Structure 6
                                +-  Structure 5   +-    (Tire)
                                |     (Wheel)   --+
  Structure 2 (Pickup Truck)  --+                  +-  Structure 7
                                |   Structure 4        (Hubcap)
                                +- (Pickup Body)
```

Structure 1 consists of the commands necessary to draw an automobile, which include the command to reference the structure for an automobile body and the command to reference the structure for a wheel, which consists of the commands to reference structures for a tire and a hubcap.

Structure 2 consists of the commands necessary to draw a pickup truck, which include the command to reference the structure for a pickup truck body and the command to reference the structure for a wheel, which consists of the commands to reference structures for a tire and a hubcap.

PDELSTRUCTNET(1, PKEEP); deletes structure 1 and structure 3. Structures 5, 6, and 7 are not deleted because they are referenced by structure 2.

PDELSTRUCTNET(2, PDELETE); deletes structures 2, 4, 5, 6, and 7, regardless of the fact that structures 5, 6, and 7 are referenced by structure 1. This means that if structure 1 is posted after the PDELSTRUCTNET call, the automobile body will appear without any wheels, since the structures for the wheels have been deleted.

9.8 PDELALLSTRUCT

PDELALLSTRUCT is used to delete all structures from memory at the same time. Any open structure is closed prior to deletion.

9.8.1 Calling Sequence

```
pdelallstruct();
```

9.8.2 Parameter Descriptions

PDELALLSTRUCT has no associated arguments.

9.8.3 Example

```
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id1 = 1;
Pint struct_id2 = 2; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id1); ...
pexecutestruct(struct_id2); ...
pclosestruct(); ...
pdelallstruct(); ...
pclosephigs(); ...
termfp();
```

9.9 PPOLYLINE3

PPOLYLINE3 is used to insert a three dimensional polyline primitive element into the currently open structure. During traversal, a connected sequence of straight lines starting at the first point specified and ending at the last point specified is generated.

9.9.1 Calling Sequence

```
ppolyline3(num_points, points);
```

9.9.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
NUM_POINTS	INPUT	Pint	An integer which specifies the number of points to be defined by the points argument.
POINTS	INPUT	Ppoint3*	The address of the first point specified in the array of points which will define the polyline.

9.9.3 Example

```
Pint num_points = 3;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 1, 1 },
                     { 0, 1, 0 } };
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ppolyline3(num_points, points); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

9.10 PPOLYLINE

PPOLYLINE inserts a two dimensional polyline primitive element into the currently open structure. During traversal, a connected sequence of straight lines starting at the first point and ending at the last point is generated.

9.10.1 Calling Sequence

```
ppolyline(num_points, points);
```

9.10.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
NUM_POINTS	INPUT	Pint	An integer which specifies the number of points to be defined by the points argument.
POINTS	INPUT	Ppoint*	The address of the first point specified in the array of points which define the polyline.

9.10.3 Example

```
Pint num_points = 3;
Ppoint points[] = { { 0, 0 },
                    { 1, 1 },
                    { 0, 1 } };
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ppolyline(num_points, points); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

9.11 PPOLYMARKER3

PPOLYMARKER3 will cause a three dimensional polymarker primitive element to be inserted into the currently open structure. During traversal, a marker is generated at each of the points specified by the POINTS argument.

9.11.1 Calling Sequence

```
ppolymarker3(num_points, points);
```

9.11.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
NUM_POINTS	INPUT	Pint	An integer which specifies the number of points to be defined by the points argument.
POINTS	INPUT	Ppoint3*	The address of the first point specified by the array of points which will define the location(s) of the polymarkers.

9.11.3 Example

```
Pint num_points = 3;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 1, 1 },
                     { 0, 1, 0 } };
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ppolymarker3(num_points, points); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

9.12 PPOLYMARKER

PPOLYMARKER causes a two dimensional polymarker primitive element to be inserted into the currently open structure. During traversal, a marker is generated at each of the points specified by the POINTS argument.

9.12.1 Calling Sequence

```
ppolymarker(num_points, points);
```

9.12.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
NUM_POINTS	INPUT	Pint	An integer which specifies the number of points to be defined by the points argument.
POINTS	INPUT	Ppoint*	The address of the first point specified by the array of points which will define the location(s) of the polymarker(s).

9.12.3 Example

```
Pint num_points = 3;
Ppoint points[] = { { 0, 0 },
                    { 1, 1 },
                    { 0, 1 } };
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ppolymarker(num_points, points); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

9.13 PTEXT3

Causes a three dimensional text primitive element to be inserted into the currently open structure. During traversal, a character string is generated on the plane specified by the arguments 'TEXT_PT' and 'DIR' relative to 'TEXT_PT'.

dir[0] specifies the vector, or path that the character string is to follow. dir[1] specifies which direction is "up" to the character string.

9.13.1 Calling Sequence

```
ptext3(text_pt, dir, text);
```

9.13.2 Parameter Descriptions

NAME	I/O		
TEXT_PT	INPUT	Ppoint3*	The point at which the first character of the text string is to be displayed.
DIR	INPUT	Pvector3*	An array of two three dimensional points. The first point (DIR[0]) specifies the direction or path which the text string is to follow. The second point (DIR[1]) specifies which direction is to be considered "up" for the text string. Both points are relative to TEXT_PT, that is, TEXT_PT is to be considered the origin when specifying DIR, regardless of TEXT_PT's actual location.
TEXT	INPUT	Pchar*	An array of text characters to be displayed in the area specified by TEXT_PT & DIR.

9.13.3 Example

```
Pchar text[] = "Display This String";
Ppoint3 text_pt = { 4, 4, 4 };
Pvector3 dir[] = { {  0, 1, 0 },
                   { -1, 0, 0 } };
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptext3(text_pt, dir, text); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

The above code specifies the text string "Display This String", which will appear starting at the three dimensional point 4, 4, 4, proceeding in a vertical direction, with the tops of the letters pointing directly to the left. By manipulating the "up" direction, the text can be rotated in three dimensional space. By manipulating the path direction, the direction of the plane upon which the text string will lie can be manipulated.

9.14 PTEXT

This routine inserts a two dimensional text primitive element into the currently open structure. During traversal, a character string is generated on the Z = 0 plane, relative to TEXT_PT.

9.14.1 Calling Sequence

```
ptext(text_pt, text);
```

9.14.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
TEXT_PT	INPUT	Ppoint*	The point at which the first character of the text string is to be displayed.
TEXT	INPUT	Pchar*	An array of text characters which are to be displayed.

9.14.3 Example

```
Pchar text[] = "Display This String";
Ppoint text_pt = { 4, 4 };
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id1); ...
ptext(text_pt, text); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

9.15 PFILLAREA3

PFILLAREA3 is used to insert a three dimensional fillarea primitive into the currently open structure. During traversal, an implicitly closed polygonal area is generated.

9.15.1 Calling Sequence

```
pfillarea3(num_points, points);
```

9.15.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
NUM_POINTS	INPUT	Pint	An integer which specifies the number of points to be defined by the points argument.
POINTS	INPUT	Ppoint3*	The address of the first point specified by the array of points which will be used to define the boundaries of the fillarea.

9.15.3 Example

```
Pint num_points = 8;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };   /*  A cube  */
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pfillarea3(num_points, points); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

9.16 PFILLAREA

PFILLAREA is used to insert a two dimensional fillarea primitive element into the currently open structure. During traversal, an implicitly closed polygonal area is generated.

9.16.1 Calling Sequence

```
pfillarea(num_points, points);
```

9.16.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
NUM_POINTS	INPUT	Pint	An integer which specifies the number of points to be defined by the points argument.
POINTS	INPUT	Ppoint*	The address of the first point specified by the array of points which will be used to define the boundaries of the fillarea.

9.16.3 Example

```
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
pclosephigs(); ...
termfp();
```

9.17 POPENWS

POPENWS adds a page to the window which is specified by the argument CONN_ID. and the workstation is identified by the integer specified for the argument WS. If a workstation WS already exists, an error is reported. POPENWS also allocates and initializes a workstation state list and a workstation description table. (NOTE: The argument WS_TYPE must be specified, but it is ignored).

9.17.1 Calling Sequence

```
popenws(ws, conn_id, ws_type);
```

9.17.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	A 4-byte signed integer that is an identifier for the work-station being opened.
CONN_ID	INPUT	Pconnid	A C-language structure which contains the unique path name of the window to which the page containing the work-station being created is added
WS_TYPE	INPUT	Pwstype	A C-language structure which also contains the unique path name of the window to which the page containing the work-station is to be added. (This argument currently is ignored)

9.17.3 Example

```
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pclosephigs(); ...
termfp();
```

9.18 PCLOSEWS

PCLOSEWS unposts any structures which are currently posted on the specified workstation, frees the workstation state list and description table, and removes the workstation page from the window in which it was opened.

9.18.1 Calling Sequence

```
pclosews(ws);
```

9.18.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	A 4-byte signed integer which acts as an identifier for the workstation which is to be closed.

9.18.3 Example

```
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.19 PPOSTSTRUCT

PPOSTSTRUCT causes the specified structure to be marked as displayed on the specified workstation. More than one structure can be posted to the same workstation. When multiple structures are posted to one workstation, they are "stacked", in the order in which they are posted, much as pages are stacked in a window. A major difference is that while a page in a window is opaque, that is, the underlying page is not visible, the pages in a workstation can be considered transparent, as if several overhead projector transparencies had been stacked on top of one another, as illustrated in the following figure:

```
        +----+
+-------+3   +--+
|1      |    |  |
|       |    |  |
|       |    |  |
|   +---+    +--+-----------+
|   |2  |    |              |
|   |   |    |              |
|   |   |    |              |
+---+   |    |              |
        |    |              |
        |    |              |
        |    |              |
    +---+    +--------------+
        |    |
        +----+
```

This figure contains 3 overlapping fillareas. In the figure, structure 1 was posted to the workstation first and therefore appears on the bottom of the stack. 2 was posted second, and appears on top of 1. 3 was posted last and appears on top.

As shown in the figure, those portions of structures which are not directly obscured by subseequently posted structures are still visible, just like a stack of overhead projector transparencies.

Posting a structure to a workstation automatically causes that structure and all referenced structures within its structure network to be traversed.

9.19.1 Calling Sequence

```
ppoststruct(ws, struct_id, priority);
```

9.19.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	A 4-byte signed integer which specifies the workstation into which the structure is to be posted.
STRUCT_ID	INPUT	Pint	A 4-byte signed integer which specifies the structure which is to be posted.
PRIORITY	INPUT	Pfloat	A float point number which specifies the priority of the structure in the stack. At the present time, PRIORITY is recorded but ignored.

9.19.3 Example

```
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
ppoststruct(ws, struct_id, priority); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.20 PUNPOSTSTRUCT

PUNPOSTSTRUCT removes the specified structure and all of its child structures from the specified workstation. The next time PREDRAWALLSTRUCT or PUPDATEWS is called, the specified structure will not appear.

9.20.1 Calling Sequence

```
punpoststruct(ws, struct_id);
```

9.20.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	A 4-byte signed integer which specifies the workstation from which the structure is to be unposted, or removed.
STRUCT_ID	INPUT	Pint	A 4-byte signed integer which specifies the structure which is to be unposted, or removed.

9.20.3 Example

```
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.21 PUNPOSTALLSTRUCT

PUNPOSTALLSTRUCT unposts (removes) all structures from the specified workstation. The structures will disappear from the screen at the next PREDRAWALLSTRUCT or PUPDATEWS.

9.21.1 Calling Sequence

```
punpostallstruct(ws);
```

9.21.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	A 4-byte signed integer which specifies the workstation from which all structures are to be unposted, or removed.

9.21.3 Example

```
Pint num_points1 = 4;
Ppoint points1[] = { { 0, 0 },
                     { 1, 0 },
                     { 1, 1 },
                     { 0, 1 } };   /*  A square  */
Pint num_points2 = 3;
Ppoint points2[] = { { 0, 0 },
                     { 2, 0 },
                     { 1, 1 } };   /*  A triangle  */
Pchar error_file[] = "filename";
Plong buf_size = 512;
Pint struct_id1 = 1;
Pint struct_id2 = 2;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id1); ...
pfillarea(num_points1, points1); ...
pclosestruct(); ...
popenstruct(struct_id2); ...
pfillarea(num_points2, points2); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
ppoststruct(ws, struct_id1, priority); ...
ppoststruct(ws, struct_id2, priority); ...
punpostallstruct(ws); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.22 PMSG

PMSG is used to display a message in the message line of the UI-PLUS screen. PMSG will move the specified message into the buffer. If the message is to be displayed immediately, follow the PMSG call with an OUTSCR call. If the specified character string is longer than 60 characters, the string will be segmented into a series of 60 character (or less) strings. When the message queue is viewed, the series of strings will be inverted. For example, if the string "This is a character string which is to be displayed in the message queue via the call PMSG. It is longer than 60 characters, so it will appear inverted in the message queue." is specified, the following will appear in the message queue:

ters, so it will appear inverted in the message queue.
essage queue via the call PMSG. It is longer than 60 charac
This is a character string which is to be displayed in the m

9.22.1 Calling Sequence

```
pmsg(ws, msg);
```

9.22.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	A 4-byte signed integer which specifies the workstation to which the message applies. At the current time, ws is recorded but ignored.
MSG	INPUT	Pchar*	The address of the array which contains the character string which is to be displayed.

9.22.3 Example

```
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.23 PSETLINETYPE

This routine is used to insert a set linetype element into the currently open structure. During traversal, this element will set the linetype entry in the traversal state list which then effects all subsequent polyline primitives, until another set linetype element is encountered.

This process is similar to the process of creating an attribute with business graphs. The difference is that attributes are tied to the primitive when using business graphs, but the "attribute", or set linetype element, is tied to the structure, not just one primitive, such that all primitives encountered in the structure use this characteristic, until a new set linetype element (if any) is encountered.

9.23.1 Calling Sequence

```
psetlinetype(linetype);
```

9.23.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
LINETYPE	INPUT	Pint	A 4-byte signed integer which specifies the linetype to apply to all subsequent polylines in the currently open structure. Available linetypes are: 1 - SOLID, 2 - DASH, 3 - DOT, 4 - DOTDASH

9.23.3 Example

```
Pint linetype = 1;    /*  Solid line  */
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetlinetype(linetype); ...
ppolyline(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.24 PSETLINECOLOURIND

This routine is used to insert a set polyline color index element into the currently open structure. During traversal, this element will set the polyline color entry in the traversal state list which then effects all subsequent polyline primitives, until a new set line color index element is encountered.

This process is similar to process of creating an attribute with business graphs. The difference is that attributes are tied to the primitive when using business graphs, but the "attribute", or, in this case, the set polyline color element, is tied to the structure, not just one primitive, such that all primitives encountered in the structure use this characteristic, until a new set polyline color element (if any) is encountered.

9.24.1 Calling Sequence

```
psetlinecolourind(index);
```

9.23.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
INDEX	INPUT	Pint	A 4-byte signed integer which specifies the color to apply to all subsequent polylines in the currently open structure. Available colors are: 0 - BLACK, 1 - RED, 2 - GREEN, 3 - YELLOW, 4 - BLUE, 5 - MAGENTA, 6 - CYAN, 7 - WHITE

9.24.3 Example

```
Pint linetype = 1;     /*  Solid line  */
Pint index = 4;        /*  Blue line   */
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetlinetype(linetype); ...
psetlinecolourind(index); ...
ppolyline(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.25 PSETMARKERTYPE

PSETMARKERTYPE inserts a set marker type element into the currently open structure. During traversal, this element will set the marker type entry in the traversal state list which will effect all subsequent marker primitives, until another set marker type element is encountered.

9.25.1 Calling Sequence

```
psetmarkertype(markertype);
```

9.25.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
MARKERTYPE	INPUT	Pint	A 4-byte signed integer which specifies the type of marker to use each time a polymarker is encountered, until a new set polymarker type element is encountered. Valid polymarker types are: 1 - POINT 2 - PLUS 3 - STAR 4 - CIRCLE 5 - CROSS

9.25.3 Example

```
Pint markertype = 2;      /*  Polymarker is a +  */
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  with a
                                      plus (+) at each
                                      corner, no lines
                                      connecting.       */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetmarkertype(markertype); ...
ppolymarker(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.26 PSETMARKERCOLOURIND

PSETMARKERCOLOURIND inserts a set polymarker color index element into the currently open structure. During traversal, this element will set the marker type entry in the traversal state list which effects subsequent marker primitives until a new set polymarker color index element is encountered.

9.26.1 Calling Sequence

```
psetmarkercolourind(index);
```

9.26.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
INDEX	INPUT	Pint	A 4-byte signed integer which specifies the color to apply to all subsequent polymarkers in the currently open structure. Available colors are: 0 - BLACK, 1 - RED, 2 - GREEN, 3 - YELLOW, 4 - BLUE, 5 - MAGENTA 6 - CYAN, 7 - WHITE

9.26.3 Example

```
Pint markertype = 2;  /*  Polymarker is a +  */
Pint index = 6;       /*  Polymarker color is magenta  */
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square with plus
                                      signs at each corner,
                                      no connecting lines,
                                      color is magenta  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetmarkertype(markertype); ...
psetmarkercolourind(index); ...
ppolymarker(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.27 PSETTEXTCOLOURIND

This routine is used to insert a set text color index element into the currently open structure. During traversal, this element will set the text color entry in the traversal state list which effects all subsequent text primitives until a new set text color index element is encountered.

9.27.1 Calling Sequence

```
psettextcolourind(index);
```

9.27.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
INDEX	INPUT	Pint	A 4-byte signed integer which specifies the color to apply to all subsequent appearances of text in the currently open structure. Available colors are: 0 - BLACK, 1 - WHITE, 2 - RED, 3 - GREEN, 4 - BLUE, 5 - CYAN, 6 - MAGENTA, 7 - YELLOW

9.27.3 Example

```
Pint index = 3;        /*  Text color is GREEN  */
Pchar text[] = "Display This String";
Ppoint text_pt = { 4, 4 };
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psettextcolourind(index); ...
ptext(text_pt, text)...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.28 PSETCHARHEIGHT

PSETCHARHEIGHT inserts a set character height element into the currently open structure. During traversal, this element will set the character height entry in the traversal state list which effects all subsequent text primitives until the next set character height element is encountered.

9.28.1 Calling Sequence

```
psetcharheight(height);
```

9.28.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
HEIGHT	INPUT	Pfloat	A floating point number which specifies the height of the text characters to be displayed. The numbers represent a scaling factor, where 1 represents a character which is the height of the display space, .10 represents one tenth the height of the display space, and so on.

9.28.3 Example

```
Pfloat height = .05    /*  5/100 of the display space  */
Pint index = 3;        /*  Text color is GREEN  */
Pchar text[] = "Display This String";
Ppoint text_pt = { 4, 4 };
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psettextcolourind(index); ...
psetcharheight(height); ...
ptext(text_pt, text)...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

2.29 PSETINTSTYLE

PSETINTSTYLE is used to insert a set interior style element into the current open structure. During traversal, this element will set the interior style entry in the traversal state list which will effect all subsequent fill area primitives until another set interior style element is encountered.

2.29.1 Calling Sequence

```
psetintstyle(style);
```

2.29.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
STYLE	INPUT	Pinterstyle	A C-language structure which contains the specifier for the type of interior to assign to all subsequent fillareas in the currently open structure. The available interior styles are: 0 - HOLLOW, 1 - SOLID, 2 - PATTERN, 3 - HATCH

2.29.3 Example

```
Pinterstyle style = 1;   /*  Interior is hollow  */
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };  /*  A square  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.30 PTRANSLATE3

PTRANSLATE3 is used to move a three dimensional structure in a workstation to another position within the workstation. This is accomplished by passing in the vector along which the structure is to be moved, which is in turn used to construct a transformation matrix. The transformation matrix is used at a later point to actually perform the move with the routines PTRANPT3, PSETLOCALTRAN3, and PSETGLOBALTRAN3.

9.30.1 Calling Sequence

```
ptranslate3(trans_vector, &err_ind, matrix);
```

9.30.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
TRANS_VECTOR	INPUT	Pvector3*	The vector along which the structure is to be moved. This is an array of three (x,y,z) coordinates.
ERR_IND	OUTPUT	Pint*	An integer which represents an error code.
MATRIX	OUTPUT	Pmatrix3	A 4 X 4 matrix which will be used when perperforming the transformation.

9.30.3 Example

```
Pint err_ind;
Pvector3 trans_vector = { 5, 5, 5 }; /*  The vector along
                                         which to move the
                                         structure  */
Pmatrix3 matrix;          /*  The transformation matrix */
Pinterstyle style = 1;    /*  Interior is hollow  */
Pint num_points = 8;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };    /*  A cube  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea3(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
ptranslate3(trans_vector, &err_ind, matrix); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.31 PTRANSLATE

PTRANSLATE is used to move a two dimensional structure in a workstation to another position within the workstation. This is accomplished by passing in the vector along which the structure is to be moved, which is in turn used to construct a transformation matrix. The transformation matrix is used at a later point to actually perform the move, with the routines PTRANPT, PSETLOCALTRAN, and PSETGLOBALTRAN.

9.31.1 Calling Sequence

```
ptranslate(trans_vector, &err_ind, matrix);
```

9.31.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
TRANS_VECTOR	INPUT	Pvector*	The vector along which the structure is to be moved. Vector is an array of X-Y coordinates.
ERR_IND	OUTPUT	Pint*	An integer which represents an error code.
MATRIX	OUTPUT	Pmatrix	A 3 X 3 matrix which will be used when performing the transformation.

9.31.3 Example

```
Pint err_ind;
Pvector trans_vector = { 5, 5 };
Pmatrix matrix;
Pinterstyle style = 1;   /*  Interior is hollow  */
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
ptranslate(trans_vector, &err_ind, matrix); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.32 PSCALE3

PSCALE3 is used to resize a three dimensional structure. This is accomplished by passing in the vector along which to resize the structure. The vector is specified in X-Y-Z coordinates. The numbers specified for the coordinates are the factors by which to multiply in order to get the new size. For example, a coordinate set of 1, 1, 1 would not change the size of the structure at all. Coordinates of 2, 1, 1 would stretch the structure to twice its width along the X-axis. Coordinates of 1, .5, 2 would shrink the structure to half its height along the Y-axis and stretch the structure to twice its depth along the Z-axis. This vector is used to construct a transformation matrix. The transformation matrix is used at a later point to actually perform the resizing, with the routines PTRANPT3, PSETLOCALTRAN3, and PSETGLOBALTRAN3.

9.32.1 Calling Sequence

```
pscale3(scale_vector, &err_ind, matrix);
```

9.32.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
SCALE_VECTOR	INPUT	Pvector3*	The vector along which the structure is to be resized. It is given as an array of X-Y-Z coordinates.
ERR_IND	OUTPUT	Pint*	An integer which represents an error code.
MATRIX	OUTPUT	Pmatrix3	A 4 X 4 matrix which will be used when perperforming the transformation.

9.32.3 Example

```
Pint err_ind;
Pvector3 trans_vector = { 5, 5, 5 }; /*  Grow the structure
                                         by a factor of 5
                                         in all directions
                                                          */
Pmatrix3 matrix;          /*  The transformation matrix  */
Pinterstyle style = 1;    /*  Interior is hollow         */
Pint num_points = 8;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };    /*  A cube  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea3(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
pscale3(trans_vector, &err_ind, matrix); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.33 PSCALE

PSCALE is used to resize a two dimensional structure. This is accomplished by passing in the vector along which to resize the structure. The vector is specified in CST coordinates. The numbers specified for the coordinates are the factors by which to multiply in order to get the new size. For example, a coordinate set of 1, 1 would not change the size of the structure at all. Coordinates of 2, 1 would stretch the structure to twice its width along the X-axis. Coordinates of 1, .5 would shrink the structure to half its height along the Y-axis. This vector is used to construct a transformation matrix. The transformation matrix is used at a later point to actually perform the resizing, with the routines PTRANPT, PSETLOCALTRAN, and PSETGLOBALTRAN.

9.33.1 Calling Sequence

```
pscale(scale_vector, &err_ind, matrix);
```

9.33.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
SCALE_VECTOR	INPUT	Pvector*	The vector along which the structure is to be resized. The vector is specified as an array of X-Y coordinates.
ERR_IND	OUTPUT	Pint*	An integer which represents an error code.
MATRIX	OUTPUT	Pmatrix	A 3 X 3 matrix which will be used when perperforming the transformation.

9.33.3 Example

```
Pint err_ind;
Pvector scale_vector = { 5, 5 };  /*  Grow the structure
                                      by a factor of 5
                                      in all directions  */
Pmatrix matrix;                   /*  The transformation
                                      matrix             */
Pinterstyle style = 1;            /*  Interior is hollow */
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
pscale(trans_vector, &err_ind, matrix); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.34 PROTATEX

PROTATEX is used to rotate a three dimensional structure around the X-axis. PROTATEX generates the three dimensional transformation matrix which will be used to perform the rotation.

9.34.1 Calling Sequence

```
protatex(angle, &err_ind, matrix);
```

9.34.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
ANGLE	INPUT	Pfloat	A float point number which specifies the angle of rotation, given in radians (there are 2 * pi radians in a circle). A positive number rotates the structure counter-clockwise, a negative number rotates it clockwise.
ERR_IND	OUTPUT	Pint*	An integer which represents an error_code.
MATRIX	OUTPUT	Pmatrix3	The transformation matrix which will be used to rotate the structure.

9.34.3 Example

```
Pint err_ind;
Pfloat angle = .5;          /*  Rotate the structure by one-
                               half of a radian (about 29
                               degrees)                    */
Pmatrix3 matrix;            /*  The transformation matrix   */
Pinterstyle style = 1;      /*  Interior is hollow          */
Pint num_points = 8;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };    /*  A cube  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
protatex(angle, &err_ind, matrix); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.35 PROTATEY

PROTATEY is used to rotate a three dimensional structure around the Y-axis. PROTATEY generates the three dimensional transformation matrix which will be used to perform the rotation.

9.35.1 Calling Sequence

```
protatey(angle, &err_ind, matrix);
```

9.35.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
ANGLE	INPUT	Pfloat	A float point number which specifies the angle of rotation, given in radians (there are 2 * pi radians in a circle). A positive number rotates the structure counter-clockwise, a negative number rotates it clockwise.
ERR_IND	OUTPUT	Pint*	An integer which represents an error_code.
MATRIX	OUTPUT	Pmatrix3	The transformation matrix which will be used to rotate the structure.

9.35.3 Example

```
Pint err_ind;
Pfloat angle = .5;        /*  Rotate the structure by one-
                             half of a radian (about 29
                             degrees)                    */
Pmatrix3 matrix;          /*  The transformation matrix   */
Pinterstyle style = 1;    /*  Interior is hollow          */
Pint num_points = 8;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };    /*  A cube  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
protatey(angle, &err_ind, matrix); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.36 PROTATEZ

PROTATEZ is used to rotate a three dimensional structure around the Z-axis. PROTATEZ generates the three dimensional transformation matrix which will be used to perform the rotation.

9.36.1 Calling Sequence

```
protatez(angle, &err_ind, matrix);
```

9.36.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
ANGLE	INPUT	Pfloat	A float point number which specifies the angle of rotation, given in radians (there are 2 * pi radians in a circle). A positive number rotates the structure counter-clockwise, a negative number rotates it clockwise.
ERR_IND	OUTPUT	Pint*	An integer which represents an error_code.
MATRIX	OUTPUT	Pmatrix3	The transformation matrix which will be used to rotate the structure.

9.36.3 Example

```
Pint err_ind;
Pfloat angle = .5;          /*  Rotate the structure by one-
                               half of a radian (about 29
                               degrees)                      */
Pmatrix3 matrix;            /*  The transformation matrix   */
Pinterstyle style = 1;      /*  Interior is hollow          */
Pint num_points = 8;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };    /*  A cube  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
protatez(angle, &err_ind, matrix); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.37 PROTATE

PROTATE is used to rotate a two dimensional structure around the origin of the axes. PROTATE generates the two dimensional transformation matrix which will be used to perform the rotation.

9.37.1 Calling Sequence

```
protate(angle, &err_ind, matrix);
```

9.37.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
ANGLE	INPUT	Pfloat	A float point number which specifies the angle of rotation, given in radians (there are 2 * pi radians in a circle). A positive number rotates the structure counter-clockwise, a negative number rotates it clockwise.
ERR_IND	OUTPUT	Pint*	An integer which represents an error_code.
MATRIX	OUTPUT	Pmatrix3	The transformation matrix which will be used to rotate the structure.

9.37.3 Example

```
Pint err_ind;
Pfloat angle = .5;          /*  Rotate the structure by one-
                               half of a radian (about 29
                               degrees)                    */
Pmatrix matrix;             /*  The transformation matrix   */
Pinterstyle style = 1;      /*  Interior is hollow          */
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } ;   /*  A square  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
protate(angle, &err_ind, matrix); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.38 PCOMPOSEMATRIX3

PCOMPOSEMATRIX3 is used to multiply three dimensional matrices together in order to combine transformations. In this manner it is possible to save both memory and CPU time.

The memory is saved when multiple matrices are composed together into one matrix, eliminating the need for multiple matrices. For example, PROTATEX produces Matrix X, PROTATEY produces Matrix Y. X and Y can then be composed back into the area reserved for Matrix X. PROTATEZ then produces its transformation matrix into the area that held Matrix Y. The new Matrix Y is then composed with the combined Matrix Y back into Matrix X.

The CPU time is saved when one transformation is now performed on the composed Matrix X, instead of a transformation on X, then one on Y, and then one on Z.

9.38.1 Calling Sequence

```
pcomposematrix3(mat_x, mat_y, &err_ind, result_matrix);
```

9.38.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
MATRIX A	INPUT	Pmatrix3	The first matrix which is to be composed with another into a resulting new matrix.
MATRIX B	INPUT	Pmatrix3	The second matrix which is to be composed with the first into a resulting new matrix.
ERR_IND	OUTPUT	Pint	An integer which represents an error code.
RESULTING MATRIX	OUTPUT	Pmatrix3	The result of the compose.

9.38.3 Example

```
Pint err_ind;
Pfloat angle_x = .5;      /*  Rotate the structure by one-
                             half of a radian (about 29
                             degrees)  (PROTATEX)        */
Pfloat angle_y = .5;      /*  Rotate the structure by one-
                             half of a radian (about 29
                             degrees)  (PROTATEY)        */
Pfloat angle_z = .5;      /*  Rotate the structure by one-
                             half of a radian (about 29
                             degrees)  (PROTATEZ)        */
Pmatrix3 matrix_x         /*  Transformation matrix for
                             PROTATEX & final result of
                             the COMPOSE                 */
Pmatrix3 matrix_y         /*  Transformation matrix for
                             PROTATEY & PROTATEZ         */
Pinterstyle style = 1;    /*  Interior is hollow          */
Pint num_points = 8;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };   /*  A cube  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
```

```
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
protatex(angle_x, &err_ind, matrix_x); ...
protatey(angle_y, &err_ind, matrix_y); ...
pcomposematrix3(matrix_x, matrix_y, &err_ind,
                matrix_x); ...
protatez(angle_z, &err_ind, matrix_y); ...
pcomposematrix3(matrix_x, matrix_y, &err_ind,
                matrix_x); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.39 PCOMPOSEMATRIX

PCOMPOSEMATRIX is used to multiply two dimensional matrices together in order to combine transformations. In this manner it is possible to save both memory and CPU time.

The memory is saved when multiple matrices are composed together into 1 matrix, eliminating the need for multiple matrices. For example, PTRANSLATE produces Matrix A for moving the structure, PSCALE produces Matrix B, for resizing the structure. A and B can then be composed back into the area reserved for Matrix A. PROTATE then produces its transformation matrix for rotating the structure back into the area that held Matrix B, the area that held the matrix resizing. The new Matrix B is then composed with the combined Matrix A back into Matrix A. The CPU time is then saved when one transformation is then all that is required to perform the move, the resize, and the rotation, instead of a transformation for each.

9.39.1 Calling Sequence

```
pcomposematrix(mat_a, mat_b, &err_ind, result_matrix);
```

9.39.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
MATRIX_X	INPUT	Pmatrix3	The first matrix which is to be composed with another into a resulting new matrix.
MATRIX_Y	INPUT	Pmatrix3	The second matrix which is to be composed with the first into a resulting new matrix.
ERR_IND	OUTPUT	Pint	An integer which represents an error code.
RESULTING MATRIX	OUTPUT	Pmatrix3	The result of the compose.

9.39.3 Example

```
Pint err_ind;
Pvector trans_vector = { 5, 5 };  /* Move the structure
                                     5 units to the right
                                     on the X-axis, 5
                                     units up vertically
                                     on the Y-axis      */
Pvector scale_vector = { 5, 5 };  /* Grow the structure by
                                     a factor of 5 in all
                                     directions         */
Pfloat angle = .5;        /* Rotate the structure by one-
                             half of a radian (about 29
                             degrees)                   */
Pmatrix matrix_a;         /* Transformation matrix for
                             PTRANSLATE & final result of
                             the COMPOSE                */
Pmatrix matrix_b;         /* Transformation matrix for
                             PSCALE and PROTATE         */
Pinterstyle style = 1;    /* Interior is hollow          */
Pint num_points = 8;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };   /*  A cube  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintstyle(style); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
```

```
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
ptranslate(trans_vector, &err_ind, matrix_a); ...
pscale(trans_vector, &err_ind, matrix_b); ...
pcomposematrix(matrix_a, matrix_b, &err_ind, matrix_a); ...
protate(angle, &err_ind, matrix_b); ...
pcomposematrix(matrix_a, matrix_b, &err_ind, matrix_a); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.40 PTRANPT3

PTRANPT3 is used to transform a three dimensional point, using a transformation matrix which has already been created. This routine is used in order to forego the use of the automatic PHIGS routines which perform the transformations for you. If this routine is used to transform a polyline which is defined with 15 points, each of the 15 points must be transformed individually.

9.40.1 Calling Sequence

```
ptranpt3(in_pt, matrix, &err_ind, out_pt);
```

9.40.2 Parameter Descriptions

NAME			
IN_PT	INPUT	Ppoint3	The coordinates of the point to be transformed.
MATRIX	INPUT	Pmatrix3	The transformation matrix to be used.
ERR_IND	OUTPUT	Pint	An integer which represents an error code.
OUT_PT	OUTPUT	Ppoint3	The coordinates of the point resulting from the transformation.

9.40.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };   /*  A cube  */

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate3(trans_vector, &err_ind, matrix_a);
for (i = 0; i < num_points;, i++)
   ptranpt3(points[i], matrix_a, &err_ind, points[i]);
ppolyline3(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.41 PTRANPT

PTRANPT is used to transform a two dimensional point, using a transformation matrix which has already been created. This routine is used in order to forego the use of the automatic PHIGS routines which perform the transformations for you. If this routine is used to transform a polyline which is defined with 15 points, each of the 15 points must be transformed individually.

9.41.1 Calling Sequence

```
ptranpt(in_pt, matrix, &err_ind, out_pt);
```

9.41.2 Parameter Descriptions

NAME			
IN_PT	INPUT	Ppoint3	The coordinates of the point to be transformed.
MATRIX	INPUT	Pmatrix3	The transformation matrix to be used.
ERR_IND	OUTPUT	Pint	An integer which represents an error code.
OUT_PT	OUTPUT	Ppoint3	The coordinates of the point resulting from the transformation.

9.41.3 Example

```
Pint err_ind, struct_id = 1, num_points = 4, ws = 1, i;
Pfloat priority = 1;
Plong buf_size = 512;
Pvector trans_vector = { 5, 5 };
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix matrix_a;
Pchar error_file[] = "filename";
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate(trans_vector, &err_ind, matrix_a);
for (i = 0; i < num_points;, i++)
   ptranpt(points[i], matrix_a, &err_ind, points[i]);
ppolyline(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.42 PBUILDTRAN3

PBUILDTRAN3 is used to build one three dimensional transformation matrix for a series of transformations, rather than just one at a time. These transformations include rotating, scaling, and translating (moving) the structure.

9.42.1 Calling Sequence

```
pbuildtran3(pt, shift, x_angle, y_angle, z_angle, scale,
            err_ind, matrix);
```

9.42.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
PT	INPUT	Ppoint3	The reference point around which rotations will be made to which the scaling will be done.
SHIFT	INPUT	Pvector3	The vector along which the structure is to be moved. It is given in 3-D coordinates.
X_ANGLE Y_ANGLE Z_ANGLE	INPUT	Pfloat	Float point numbers which specify the angle of rotation, given in radians (there are 2 * pi radians in a circle). A positive number rotates the structure counter-clockwise, and a negative number rotates it clockwise.

SCALE	INPUT	Pvector3	The vector along which the structure is to be resized. Scale is specified as an array of X-Y-Z coordinates.
ERR_IND	OUTPUT	Pint	An integer which represents an error code.
MATRIX	OUTPUT	Pmatrix3	The transformation matrix which is generated.

9.42.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat x_angle = .5, y_angle = .25, z_angle = 0.;
Pfloat priority = 1;
Pvector3 scale = { 10, 10, 10 };
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };  /*  A cube  */
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pbuildtran3(points[0], trans_vector, x_angle, y_angle,
            z_angle, scale, &err_ind, matrix_a);
for (i = 0; i < num_points;, i++)
   ptranpt3(points[i], matrix_a, &err_ind, points[i]);
ppolyline3(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.43 PBUILDTRAN

PBUILDTRAN is used to build one two dimensional transformation matrix for a series of transformations, rather than just one at a time. These transformations include rotating, scaling, and translating (moving) the structure.

9.43.1 Calling Sequence

```
pbuildtran(pt, shift, angle, scale, &err_ind, matrix);
```

9.43.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
PT	INPUT	Ppoint	The reference point around which rotations will be made to which the scaling will be done.
SHIFT	INPUT	Pvector	The vector along which the structure is to be moved. It is given in 2-D coordinates.
ANGLE	INPUT	Pfloat	A float point number which specifies the angle of rotation, given in radians (there are 2 * pi radians in a circle). A positive number rotates the structure counter-clockwise, and a negative number rotates it clockwise.
SCALE	INPUT	Pvector	The vector along which the structure is to be resized. Scale is specified as an array of X-Y coordinates.
ERR_IND	OUTPUT	Pint	An integer which represents an error code.
MATRIX	OUTPUT	Pmatrix	The transformation matrix which is generated.

9.43.3 Example

```
Pint err_ind, struct_id = 1, num_points = 4, ws = 1, i;
Pfloat x_angle = .5, y_angle = .25;
Pfloat priority = 1;
Pvector scale = { 10, 10 };
Pvector trans_vector = { 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix matrix_a;
Pchar error_file[] = "filename";
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
pbuildtran(points[0], trans_vector, x_angle, y_angle,
          scale, &err_ind, matrix_a);
for (i = 0; i < num_points;, i++)
   ptranpt(points[i], matrix_a, &err_ind, pcints[i]);
ppolyline(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.44 PCOMPOSETRAN3

PCOMPOSETRAN3 is used to build one three dimensional transformation matrix for a series of transformations, and compose that matrix together with the specified matrix. These transformations include rotating, scaling, and translating (moving) the structure.

9.44.1 Calling Sequence

```
pcomposetran3(matrix_a, pt, shift, x_angle, y_angle,
              z_angle, scale, &err_ind, matrix_b);
```

9.44.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
MATRIX_A	INPUT	Pmatrix3	The initial matrix with which to compose the transformations specified.
PT	INPUT	Ppoint3	The reference point around which rotations will be made to which the scaling will be done.
SHIFT	INPUT	Pvector3	The vector along which the structure is to be moved. It is given in 3-D coordinates.
X_ANGLE Y_ANGLE Z_ANGLE	INPUT	Pfloat	Float point numbers which specify the angle of rotation, given in radians (there are 2 * pi radians in a circle). A positive number rotates the structure counter-clockwise, and a negative number rotates it clockwise.

SCALE	INPUT	Pvector3	The vector along which the structure is to be resized. Scale is specifieu as an array of X-Y-Z coordinates.
ERR_IND	OUTPUT	Pint	An integer which represents an error code.
MATRIX_B	OUTPUT	Pmatrix3	The transformation matrix which is generated.

9.44.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat x_angle = .5, y_angle = .25, z_angle = 0.;
Pfloat priority = 1;
Pvector3 scale = { 10, 10, 10 };
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };   /*  A cube  */
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate3(trans_vector, &err_ind, matrix_a);
pcomposetran3(matrix_a, points[0], trans_vector, x_angle,
             y_angle, z_angle, scale, &err_ind, matrix_a);
for (i = 0; i < num_points;, i++)
   ptranpt3(points[i], matrix_a, &err_ind, points[i]);
ppolyline3(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.45 PCOMPOSETRAN

PCOMPOSETRAN is used to build one two dimensional transformation matrix for a series of transformations and compose it together with a specified matrix. These transformations include rotating, scaling, and translating (moving) the structure.

9.45.1 Calling Sequence

```
pcomposetran(matrix_a, pt, shift, angle, scale, &err_ind,
             matrix_b);
```

9.45.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
MATRIX_A	INPUT	Pmatrix	The initial matrix with which the specified transformations will be composed.
PT	INPUT	Ppoint	The reference point around which rotations will be made to which the scaling will be done.
SHIFT	INPUT	Pvector	The vector along which the structure is to be moved. It is given in 2-D coordinates.
ANGLE	INPUT	Pfloat	A float point number which specifies the angle of rotation, given in radians (there are 2 * pi radians in a circle). A positive number rotates the structure counter-clockwise, and a negative number rotates it clockwise.

SCALE	INPUT	Pvector	The vector along which the structure is to be resized. Scale is specified as an array of X-Y coordinates.
ERR_IND	OUTPUT	Pint	An integer which represents an error code.
MATRIX_B	OUTPUT	Pmatrix	The transformation matrix which is generated.

9.45.3 Example

```
Pint err_ind, struct_id = 1, num_points = 4, ws = 1, i;
Pfloat x_angle = .5, y_angle = .25;
Pfloat priority = 1;
Pvector scale = { 10, 10 };
Pvector trans_vector = { 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix matrix_a;
Pchar error_file[] = "filename";
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate(trans_vector, &err_ind, matrix_a);
pcomposetran(matrix_a, points[0], trans_vector, x_angle,
             y_angle, scale, &err_ind, matrix_a);
for (i = 0; i < num_points;, i++)
   ptranpt(points[i], matrix_a, &err_ind, points[i]);
ppolyline(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.46 PSETINTCOLOURIND

PSETINTCOLOURIND inserts a set interior color index element into the currently open structure. During traversal, this element will set the interior color index entry in the traversal state list which will effect all subsequent fill area primitives until a new PSETINTCOLOURIND is encountered.

9.46.1 Calling Sequence

```
psetintcolourind(index);
```

9.46.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
INDEX	INPUT	Pint	The number of the color to apply to all subsequent fill areas. Currently, valid colors are: 0 - BLACK. 1 - RED, 2 - GREEN, 3 - YELLOW, 4 - BLUE, 5 - MAGENTA, 6 - CYAN, 7 - WHITE

9.46.3 Example

```
Pint index = 5;          /*  Interior is magenta  */
Pint num_points = 4;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };  /*  A square  */
Pchar error_file[] = "filename";
Pchar msg[] = "Your message here!";
Plong buf_size = 512;
Pint struct_id = 1;
Pint ws = 1;
Pconnid conn_id = "w1.f1.w2";
Pfloat priority = 1;
Pwstype ws_type = "w1.f1.w2"; ...
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
psetintcolourind(index); ...
pfillarea(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, ws_type); ...
pmsg(ws, msg); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.47 PSETLOCALTRAN3

PSETLOCALTRAN3 inserts a three dimensional set local transformation element into the currently open structure. During traversal, this element will modify the local transformation entry in the traversal state list by composing the specified transformation with the current entry in the specified manner, which will effect all subsequent primitives, until a new PSETLOCALTRAN3 is encountered.

9.47.1 Calling Sequence

```
psetlocaltran3(matrix, comp_type);
```

9.47.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
MATRIX	INPUT	Pmatrix3	The transformation matrix to use when the transformation is performed.
COMP_TYPE	INPUT	Pcomptype	The type of compose to perform. Valid comptypes are: 0 - PREPLACE - replace the local transformation matrix currently in place with the specified matrix. 1 - PPOSTCONCATENATE - Perform the local transformation currently in place, then perform the transformation specified with the new matrix. 2 - PPRECONCATENATE - Perform the transformation specified by MATRIX, then perform the transformation that was already in place.

9.47.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat x_angle = .5, y_angle = .25, z_angle = 0.;
Pfloat priority = 1;
Pvector3 scale = { 10, 10, 10 };
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Pcomptype comp_type = PREPLACE, ctype = PPRECONCATENATE;
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 },
                     { 0, 0, 1 },
                     { 1, 0, 1 },
                     { 1, 1, 1 },
                     { 0, 1, 1 } };   /*  A cube  */

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate3(trans_vector, &err_ind, matrix_a);
psetlocaltran3(matrix_a, ctype);
pfillarea3(num_points, points);
pcomposetran3(matrix_a, points[0], trans_vector, x_angle,
             y_angle, z_angle, scale, &err_ind, matrix_a);
psetlocaltran3(matrix_a, comp_type);
ppolyline3(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.48 PSETLOCALTRAN

PSETLOCALTRAN inserts a two dimensional set local transformation element into the currently open structure. During traversal, this element will modify the local transformation entry in the traversal state list by composing the specified transformation with the current entry in the specified manner, which will effect all subsequent primitives, until a new PSETLOCALTRAN is encountered.

9.48.1 Calling Sequence

```
psetlocaltran(matrix, comp_type);
```

9.48.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
MATRIX	INPUT	Pmatrix	The transformation matrix to use when the transformation is performed.
COMP_TYPE	INPUT	Pcomptype	The type of compose to perform. Valid comptypes are: 0 - PREPLACE - replace the local transformation matrix currently in place with the specified matrix. 1 - PPOSTCONCATENATE - Perform the local transformation currently in place, then perform the transformation specified with the new matrix. 2 - PPRECONCATENATE - Perform the transformation specified by MATRIX, then perform the transformation that was already in place.

9.48.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat x_angle = .5, y_angle = .25;
Pfloat priority = 1;
Pvector scale = { 10, 10 };
Pvector trans_vector = { 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix matrix_a;
Pchar error_file[] = "filename";
Pcomptype comp_type = PREPLACE, ctype = PPRECONCATENATE;
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate(trans_vector, &err_ind, matrix_a);
psetlocaltran(matrix_a, ctype);
pfillarea(num_points, points);
pcomposetran(matrix_a, points[0], trans_vector, x_angle,
            y_angle, scale, &err_ind, matrix_a);
psetlocaltran(matrix_a, comp_type);
ppolyline(num_points, points); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.49 PSETGLOBALTRAN3

PSETGLOBALTRAN3 inserts a three dimensional set global transformation element into the currently open structure. During traversal, this element will modify the global transformation entry in the traversal state list by composing the specified transformation with the current entry in the specified manner, which will effect all subsequent primitives, until a new PSETGLOBALTRAN3 is encountered.

9.49.1 Calling Sequence

```
psetglobaltran3(matrix);
```

9.49.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
MATRIX	INPUT	Pmatrix3	The transformation matrix to use when the transformation is performed.

9.49.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 }  };   /*  A square  */

Ppoint3 pts[] = { { 2, 2, 2 },
                  { 3, 3, 3 },
                  { 4, 4, 4 },
                  { 5, 5, 5 } };  /*  A line  */
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate3(trans_vector, &err_ind, matrix_a);
pglobaltran3(matrix_a);
pfillarea3(num_points, points);
ppolyline3(num_points, pts); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.50 PSETGLOBALTRAN

PSETGLOBALTRAN inserts a two dimensional set global transformation element into the currently open structure. During traversal, this element will modify the global transformation entry in the traversal state list by composing the specified transformation with the current entry in the specified manner, which will effect all subsequent primitives, until a new PSETGLOBALTRAN is encountered.

9.50.1 Calling Sequence

```
psetglobaltran(matrix);
```

9.50.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
MATRIX	INPUT	Pmatrix	The transformation matrix to use when the transformation is performed.

9.50.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector trans_vector = { 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix matrix_a;
Pchar error_file[] = "filename";
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };    /*  A square  */
Ppoint pts[] = { { 2, 2 },
                 { 3, 3 },
                 { 4, 4 },
                 { 5, 5 } };  /*  A line  */
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate(trans_vector, &err_ind, matrix_a);
pglobaltran(matrix_a);
pfillarea(num_points, points);
ppolyline(num_points, pts); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.51 PSETWSWINDOW3

PSETWSWINDOW3 sets the three dimensional workstation window in the workstation state list, in normalized projection coordinates. This routine sets up the area of the picture that is actually going to be viewed on the workstation.

9.51.1 Calling Sequence

```
psetwswindow3(ws, window);
```

9.51.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	An integer which identifies the workstation in which the picture is to be viewed.
WINDOW	INPUT	Plimit3	The coordinates which define which part of the picture to display.

NOTE: The coordinates which define the area of the picture to display are normalized projection coordinates. They are real numbers between 0 and 1 which define the specific area of the picture that is to be displayed on the screen.

Consider the following figure:

```
1 +----------------+
  |                |
  |      +---+     |
  |  +-+ |   |     |
  |  | +-+   |     |
  |  +-------+     |
  |                |
0 +----------------+
  0                1
```

Figure A

```
+---------------------------+
|  +--------------------+   |
|  |                    |   |
|  |                    |   |
|  |                    |   |
|  |      Viewport      |   |
|  |                    |   |
|  |                    |   |
|  |                    |   |
|  +--------------------+   |
+------- Workstation -------+
```

Figure B

In Figure A, the picture which is to be displayed on the workstation is shown. (It should be noted that there is also a Z-axis in Figure A which is not visible). In Figure B, the workstation and the viewport on the workstation in which the picture is to appear is shown. The fillarea shown in Figure A will be displayed inside of the viewport in Figure B. PSETWSWINDOW3 is used to determine how much of the picture in Figure A will actually be visible inside of the viewport in Figure B. For example, if the area shown in Figure C below is displayed in the viewport in Figure B, the result will be Figure D:

```
1   +----------------+
    |                |
    |        +---+   |
    |    +-+ |   |   |
,5  + - -+-  + - + -+|
    |    |       |   |
    |    +-------+   |
    |                |
0   +----------------++
    0       .5       1
```

Figure C

```
+---------------------------+
|  +--------------------+   |
|  |                    |   |
|  |                    |   |
|  |                    |   |
|  |     |      |       |   |
|  |     +------+       |   |
|  |                    |   |
|  |                    |   |
|  +--------------------+   |
+------- Workstation ------+
```

Figure D

The way in which the area shown in Figure C is specified is as follows:

```
psetwswindow3(ws, window);
```

where ws is equal to the id number for the window the picture is to be displayed in, and window is the set of normalized projection coordinates which specify which part of the picture to display, in this case, 0, .9, 0, .5, 0, 0. The coordinates specified are, in order, the minimum number on the X-axis to display, the maximum number on the X-axis to display, the minimum number on the Y-axis to display, the maximum number on the Y-axis to display, the minimum number on the Z-axis to display, and the maximum number on the Z-axis to display.

9.51.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Ppoint3  points[] = { { 0, 0, 0 },
                      { 1, 0, 0 },
                      { 1, 1, 0 },
                      { 0, 1, 0 } };   /*  A square  */
Plimit3 window = { 0.0, 0.5, 0.0, 0.5, 0.0, 0.5 };
Ppoint3 pts[] = { { 2, 2, 2 },
                  { 3, 3, 3 },
                  { 4, 4, 4 },
                  { 5, 5, 5 } };   /*  A line  */

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate3(trans_vector, &err_ind, matrix_a);
pglobaltran3(matrix_a);
pfillarea3(num_points, points);
ppolyline3(num_points, pts); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
psetwindow3(ws, window);
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.52 PSETWSWINDOW

PSETWSWINDOW sets the two dimensional workstation window in the workstation state list, in normalized projection coordinates. This routine sets up the area of the picture that is actually going to be viewed on the workstation.

9.52.1 Calling Sequence

```
psetwswindow(ws, window);
```

9.52.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	An integer which identifies the workstation in which the picture is to be viewed.
WINDOW	INPUT	Plimit	The coordinates which define which part of the picture to display.

NOTE: The coordinates which define the area of the picture to display are normalized projection coordinates. They are real numbers between 0 and 1 which define the specific area of the picture that is to be displayed on the screen.

Consider the following figure:

```
1 +-----------------+
  |                 |
  |       +---+     |
  |   +-+ |   |     |
  |   | +-+   |     |
  |   |       |     |
  |   +-------+     |
  |                 |
0 +-----------------+
  0                 1
```

Figure A

```
+---------------------------+
|  +---------------------+  |
|  |                     |  |
|  |                     |  |
|  |                     |  |
|  |      Viewport       |  |
|  |                     |  |
|  |                     |  |
|  |                     |  |
|  +---------------------+  |
+------- Workstation -------+
```

Figure B

In Figure A, the picture which is to be displayed on the workstation is shown. In Figure B, the workstation and the viewport on the workstation in which the picture is to appear is shown. The fillarea shown in Figure A will be displayed inside of the viewport in Figure B. PSETWSWINDOW is used to determine how much of the picture in Figure A will actually be visible inside of the viewport in Figure B. For example, if the area shown in Figure C below is displayed in the viewport in Figure B, the result will be Figure D:

```
1  +----------------+
   |                |
   |        +---+   |
   |     +-+ |   |  |
,5 + - -+-  + - + -+|
   |     |       |  |
   |     +-------+  |
   |                |
0  +----------------++
   0       .5       1
```

Figure C

```
+-------------------------+
|  +--------------------+ |
|  |                    | |
|  |                    | |
|  |      |     |       | |
|  |      +-----+       | |
|  |                    | |
|  |                    | |
|  +--------------------+ |
+------- Workstation ------+
```

Figure D

The way in which the area shown in Figure C is specified is as follows:

```
psetwswindow(ws, window);
```

where ws is equal to the id number for the window in which to display the picture, and window is the set of normalized projection coordinates which specify which part of the picture to display, in this case, 0, .9, 0, .5. The coordinates specified are, in order, the minimum number on the X-axis to display, the maximum number on the X-axis to display, the minimum number on the Y-axis to display, and the maximum number on the Y-axis to display.

9.52.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector trans_vector = { 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix matrix_a;
Pchar error_file[] = "filename";
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };   /*  A square  */
Plimit window = { 0.0, 0.5, 0.0, 0.5 };
Ppoint pts[] = { { 2, 2 },
                 { 3, 3 },
                 { 4, 4 },
                 { 5, 5 } };   /*  A line  */

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate(trans_vector, &err_ind, matrix_a);
pglobaltran(matrix_a);
pfillarea(num_points, points);
ppolyline(num_points, pts); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
psetwindow(ws, window);
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.53 PSETWSVIEWPORT3

PSETWSVIEWPORT3 is used to set the three dimensional workstation viewport size in the workstation state list. In order to use this routine, PINQDISPLAYSPACESIZE3 must first be called in order to determine the size of the available display space. This will return the length of the X-axis, the Y-axis, and the Z-axis. The viewport is then specified in real number coordinates from 0 to the length of the axis as returned from PINQDISPLAYSPACESIZE3 for each axis.

9.53.1 Calling Sequence

```
iipsetwsviewport3(ws, viewport);
```

9.53.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	The integer that serves as the identifier for the workstation whose viewport is to be set.
VIEWPORT	INPUT	Plimit3	The viewport limits.

9.53.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 } };   /*  A square  */
Plimit3 window = { 0.0, 0.5, 0.0, 0.5, 0.0, 0.5 };
Ppoint3 pts[] = { { 2, 2, 2 },
                  { 3, 3, 3 },
                  { 4, 4, 4 },
                  { 5, 5, 5 } };  /*  A line  */
Pdspsize3 size;
Plimit3 limit;

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate3(trans_vector, &err_ind, matrix_a);
pglobaltran3(matrix_a);
pfillarea3(num_points, points);
ppolyline3(num_points, pts); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
psetwindow3(ws, window);
pinqdisplayspacesize3(wstyp, &err_ind, &size);
limit.xmin = limit.ymin = limit.zmin = 0.0;
limit.xmax = size.device.x / 2.0;
limit.ymax = size.device.y / 2.0;
limit.zmax = size.device.z / 2.0;
psetwsviewport3(ws, limit);
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.54 PSETWSVIEWPORT

PSETWSVIEWPORT is used to set the two dimensional workstation viewport size in the workstation state list. In order to use this routine, PINQDISPLAYSPACESIZE must first be called in order to determine the size of the available display space. This will return the length of the X-axis and the Y-axis. Viewport is then specified in real number coordinates from 0 to the length of the axis as returned from PINQDISPLAYSPACESIZE for each axis.

9.54.1 Calling Sequence

```
psetwsviewport(ws, viewport);
```

9.54.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	The integer that serves as the identifier for the workstation whose viewport is to be set.
VIEWPORT	INPUT	Plimit	The viewport limits.

9.54.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector trans_vector = { 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix matrix_a;
Pchar error_file[] = "filename";
Ppoint points[] = { { 0, 0 },
                    { 1, 0 },
                    { 1, 1 },
                    { 0, 1 } };
Plimit window = { 0.0, 0.5, 0.0, 0.5 };
Ppoint pts[] = { { 2, 2 },
                 { 3, 3 },
                 { 4, 4 },
                 { 5, 5 } };
Pdspsize size;
Plimit limit;

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate(trans_vector, &err_ind, matrix_a);
pglobaltran(matrix_a);
pfillarea(num_points, points);
ppolyline(num_points, pts); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
psetwindow(ws, window);
pinqdisplaysize(wstyp, &err_ind, size);
limit.xmin  = limit.ymin = 0.0;
limit.xmax = size.device.x / 2.0;
limit.ymax = size.device.y / 2.0;
psetwsviewport(ws, limit);
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.55 PREDRAWALLSTRUCT

PREDRAWALLSTRUCT calls OUTSCR and causes all structures in a workstation to be re-drawn.

9.55.1 Calling Sequence

```
predrawallstruct(ws, control_flag);
```

9.55.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	The integer that serves as the identifier for the workstation in which all structures are to be redrawn.
CONTROL_ FLAG	INPUT	Pcontrol	A flag which specifies if the structures are always to be redrawn, or if they are to be redrawn only if they have changed. (This argument is currently ignored.)

9.55.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 } };
Ppoint3 pts[] = { { 2, 2, 2 },
                  { 3, 3, 3 },
                  { 4, 4, 4 },
                  { 5, 5, 5 } };
Pcontrol contrl = PALWAYS;
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate3(trans_vector, &err_ind, matrix_a);
pglobaltran3(matrix_a);
pfillarea3(num_points, points);
ppolyline3(num_points, pts); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
predrawallstruct(ws, contrl);
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.56 PUPDATEWS

PUPDATEWS is used to re-draw the workstation. This routine calls OUTSCR.

9.56.1 Calling Sequence

```
pupdatews(ws, regen_flag);
```

9.56.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	The integer that serves as the identifier for the workstation which is to be updated.
REGEN_ FLAG	INPUT	Pregen	A flag which specifies under circumstances the workstation is to be updated. (This argument is currently ignored).

9.56.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Ppoint3 points[] = { { 0, 0, 0 },
                     { 1, 0, 0 },
                     { 1, 1, 0 },
                     { 0, 1, 0 } };   /*  A square  */
Ppoint3 pts[] = { { 2, 2, 2 },
                  { 3, 3, 3 },
                  { 4, 4, 4 },
                  { 5, 5, 5 } };   /*  A line  */
Pregen contrl = PPERFORM;
```

```
initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate3(trans_vector, &err_ind, matrix_a);
pglobaltran3(matrix_a);
pfillarea3(num_points, points);
ppolyline3(num_points, pts); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
ppoststruct(ws, struct_id, priority); ...
pupdatews(ws, contrl);
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.57 PINQWSTYPES

PINQWSTYPES returns the list of workstation types which can be used as parameters to POPENWS.

9.57.1 Calling Sequence

```
pinqwstypes(size, &err_ind, buffer, types, &tot_size);
```

9.57.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
SIZE	INPUT	Pint	The size of the buffer.
ERR_IND	OUTPUT	Pint*	An integer that represents an error code.
BUFFER	OUTPUT	Pchar*	The output buffer.
TYPES	OUTPUT	Pwstypelst	Fixed information.
TOT_SIZE	OUTPUT	Pint*	The size of the buffer space actually used.

9.57.3 Example

```
Pint err_ind, size = 1024, total;
Pwstypelst types;
Pchar buffer[size];
Plong buf_size = 512;
Pchar error_file[] = "filename";

initfp(); ...
popenphigs(error_file, buf_size); ...
pinqwstypes(size, &err_ind, buffer, types, &total);
pclosephigs(); ...
termfp();
```

9.58 PINQWSCONNTYPE

PINQWSCONNTYPE returns the connection identifier and specific workstation type associated with the specified open workstation.

9.58.1 Calling Sequence

```
pinqwsconntype(ws, size, &err_ind, buffer, conn_id,
               tot_size, ws_type);
```

9.58.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
WS	INPUT	Pint	The workstation identifier.
SIZE	INPUT	Pint	The size of the buffer.
ERR_IND	OUTPUT	Pint*	An integer that represents an error code.
BUFFER	OUTPUT	Pchar*	The output buffer.
CONN_ID	OUTPUT	Pconnid	The connection id.
TOT_SIZE	OUTPUT	Pint*	The size of the buffer space actually used.
WS_TYPE	OUTPUT	Pwstype	Fixed information.

9.58.3 Example

```
Pint ws = 1;
Pint err_ind, size = 1024, total;
Pwstypelst types;
Pchar buffer[size];
Plong buf_size;
Pchar error_file[] = "filename";
Pconnid conn_id = "w1.f1.w2", id;
Pwstype wstyp = "w1.f1.w2", type;
```

```
initfp();
popenphigs(error_file, buf_size); ...
popenws(ws, conn_id, wstyp); ...
Pinqwsconntype(ws, size, &err_ind, buffer, id, &total,
               type);
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.59 PINQDISPLAYSPACESIZE3

This routine returns the three dimensional display size of the specified workstation.

9.59.1 Calling Sequence

```
pinqdisplayspacesize3(type, &err_ind, size);
```

9.59.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
TYPE	INPUT	Pint	Workstation type.
ERR_IND	OUTPUT	Pint	Error indicator.
SIZE	OUTPUT	Pdspsize3	Display size.

9.59.3 Example

```
Pint err_ind, struct_id = 1, num_points = 8, ws = 1, i;
Pfloat priority = 1;
Pvector3 trans_vector = { 5, 5, 5 };
Plong buf_size = 512;
Pconnid conn_id = "w1.f1.w2";
Pwstype wstyp = "w1.f1.w2";
Pmatrix3 matrix_a;
Pchar error_file[] = "filename";
Ppoint3  points[] = { { 0, 0, 0 },
                      { 1, 0, 0 },
                      { 1, 1, 0 },
                      { 0, 1, 0 } };
Plimit3 window = { 0.0, 0.5, 0.0, 0.5, 0.0, 0.5 };
Ppoint3 pts[] = { { 2, 2, 2 },
                  { 3, 3, 3 },
                  { 4, 4, 4 },
                  { 5, 5, 5 } };   /*  A line  */
Pdspsize3 size;

initfp(); ...
popenphigs(error_file, buf_size); ...
popenstruct(struct_id); ...
ptranslate3(trans_vector, &err_ind, matrix_a);
pglobaltran3(matrix_a);
pfillarea3(num_points, points);
ppolyline3(num_points, pts); ...
pclosestruct(); ...
popenws(ws, conn_id, wstyp); ...
psetwindow3(ws, window);
pinqdisplayspacesize3(wstyp, &err_ind, &size);
ppoststruct(ws, struct_id, priority); ...
punpoststruct(ws, struct_id); ...
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.60 PINQDISPLAYSPACESIZE

This routine returns the two dimensional display size of the specified workstation.

9.60.1 Calling Sequence

```
pinqdisplayspacesize(type, err_ind, size);
```

9.60.2 Parameter Descriptions

NAME	I/O	TYPE	DESCRIPTION
TYPE	INPUT	Pint	Workstation type. Can be determined by calling PINQWSCONNTYPE.
ERR_IND	OUTPUT	Pint	Error indicator.
SIZE	OUTPUT	Pdspsize	Display size.

9.60.3 Example

```
Pint ws = 1;
Pint err_ind, size = 1024, total;
Pwstypelst types;
Pchar buffer[size];
Plong buf_size = 512;
Pchar error_file[] = "filename";
Pconnid conn_id = "w1.f1.w2", id;
Pwstype wstyp = "w1.f1.w2", type;

initfp(); ...
popenphigs(error_file, buf_size); ...
popenws(ws, conn_id, wstyp); ...
pinqwsconntype(ws, size, &err_ind, buffer, id, &total,
               type);
pclosews(ws); ...
pclosephigs(); ...
termfp();
```

9.61 PHIGS Data Structures

This section lists the PHIGS data structures necessary for the PHIGS callable routines. These structures are included in the include file ELEMENT.H.

```
typedef char   Pchar;     /*  Phigs CHARacter                 */
typedef int    Pint;      /*  Phigs INTeger                   */
typedef long   Plong;     /*  Phigs LONG integer              */
typedef float  Pfloat;    /*  FLOATing point data             */
typedef int    Ptype;     /*  Phigs element TYPE declaration */

typedef struct                                  /*  Integer POINT  */
   {
   Pint x;        /*  X coordinate  */
   Pint y;        /*  Y coordinate  */
   }Pipoint;

typedef struct                              /*  3-d Integer POINT  */
   {
   Pint x;        /*  X coordinate  */
   Pint y;        /*  Y coordinate  */
   Pint z;        /*  Z coordinate  */
   }Pipoint3;

typedef enum                    /*  CONTROL parameter (for redraw
                              structures)  */
   {
   PCONDITIONALLY,
   PALWAYS
   }Pcontrol;

typedef struct
   {
   Pfloat xmin;
   Pfloat xmax;
   Pfloat ymin;
   Pfloat ymax;
   }Plimit;

typedef struct
   {
   Pfloat xmin;
   Pfloat xmax;
   Pfloat ymin;
   Pfloat ymax;
   Pfloat zmin;
   Pfloat zmax;
   }Plimit3;
```

```
typedef enum
   {
   PPERFORM,
   PPOSTPONE
   }Pregen;

typedef EPATH Pconnid;        /*  connection identifier          */
typedef EPATH Pwstype;        /*  WorkStation TYPE (window name) */

typedef struct
     {
     Pint number;             /*  number of WS types in list     */
     Pwstype *ws_types;       /*  list of types                  */
     }Pwstypelst;

typedef Pfloat Pmatrix[3][3];     /*  2-D space transformation
                                  matrix */
typedef Pfloat Pmatrix3[4][4];    /*  3-D space transformation
                                  matrix */

typedef enum      /*  matrix COMPosition type  */
   {
   PPRECONCATENATE,
   PPOSTCONCATENATE,
   PREPLACE
   }Pcomptype;

typedef struct    /*  2-D point  */
   {
   Pfloat x;
   Pfloat y;
   }Ppoint;

typedef struct    /*  3-D point  */
   {
   Pfloat x;
   Pfloat y;
   Pfloat z;
   }Ppoint3;

typedef Ppoint  Pvector;      /*  2-D vector definition  */
typedef Ppoint3 Pvector3;     /*  3-D vector definition  */

typedef enum
   {
   PDC_METERS,
   PDC_OTHER
   }Pdevunits;
```

```
typedef struct
   {
   Pdevunits units;                          /*  Device coordinate
                        units  */
   Ppoint3   device;                  /*  device volume in coordinate
                        units  */
   Pipoint3  raster;                     /*  device volume in raster
                        units  */
   }Pdspsize3;

typedef struct
   {
   Pdevunits units;                          /*  Device coordinate
                        units  */
   Ppoint    device;                    /*  device size in coordinate
                        units  */
   Pipoint   raster;                       /*  device size in raster
                        units  */
   }Pdspsize;

typedef enum     /*  REFerence Flag (delete structure
                     network)  */
   {
   PDELETE,
   PKEEP
   }Preff;

typedef enum     /*  interior style for fill area  */
     {
     PHOLLOW,
     PSOLID,
     PPATTERN,
     PHATCH,
     PEMPTY
     }Pinterst;

#define Pinterstyle Pinterst

/*  line types  */

#define PLN_SOLID   1
#define PLN_DASH    2
#define PLN_DOT     3
#define PLN_DOTDASH 4
```

```
/*  marker types  */

#define PMK_POINT   1
#define PMK_PLUS    2
#define PMK_STAR    3
#define PMK_O       4
#define PMK_X       5

/*  available colors  */

#define COL_BLACK   0
#define COL_RED     1
#define COL_GREEN   2
#define COL_YELLOW  3
#define COL_BLUE    4
#define COL_MAGENTA 5
#define COL_CYAN    6
#define COL_WHITE   7
```

☆U.S. Government Printing Office 1992—648-127/62200