INTERNATIONAL STANDARD

ISO/IEC 9541-3

First edition
1994-05-01

Information technology — Font information interchange —

Part 3:
Glyph shape representation

Technologies de l'information — Échange d'informations sur les fontes —

Partie 3: Représentation de la forme de glyphes

Reference number
ISO/IEC 9541-3:1994(E)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Contents

Page

Section 1 General **1**

1.1 Scope **1**

1.2 Conformance **1**

1.3 Normative references **1**

1.4 Notation **2**

1.5 Overview of glyph shape representation **2**

1.6 Specification of glyph shapes (GSHAPES) **3**

1.7 Extensions to the font interchange format **3**

1.7.1 ASN.1 ..... **3**

1.7.2 SGML **3**

Section 2 Type 1 glyph shape representation **4**

2.1 Scope **4**

2.2 Definitions **4**

2.3 Overview of Type 1 glyph shape representation architecture **5**

2.4 Type 1 glyph shape concepts **5**

2.4.1 Glyph coordinate system **5**

2.4.2 Glyph procedure language **6**

2.4.3 Glyph procedure interpreter **6**

2.4.4 Alignment position **6**

2.4.5 Flat position **6**

2.4.6 Overshoot position **6**

2.4.7 Overshoot suppression **6**

2.4.8 Alignment zone **6**

2.4.9 Hints **7**

2.4.10 Hint zone **9**

© ISO/IEC 1994
All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case Postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.4.11 Path direction **9**

2.4.12 Reference point **9**

2.4.13 Flex mechanism **10**

2.4.14 Hint substitution **10**

2.4.15 Bézier curve **10**

2.5 Glyph procedure interpreter model **11**

2.6 Type 1 shape properties (T1SHAPES) **11**

2.6.1 General properties (T1GENERAL) **11**

2.6.1.1 Password (PASSWORD) **12**

2.6.1.2 Paint type (PAINTTYPE) **12**

2.6.1.3 Unique identifier (UNIQUEID) **12**

2.6.2 Typographic color properties (T1COLOR) **13**

2.6.2.1 Top alignment zones (BLUEVALUES) **13**

2.6.2.2 Bottom alignment zones (OTHERBLUES) **14**

2.6.2.3 Family top alignment (FAMILYBLUES) **14**

2.6.2.4 Family bottom alignment zones (FAMILYOTHERBLUES) **15**

2.6.2.5 Overshoot suppression size (BLUESCALE) **16**

2.6.2.6 Overshoot enforcement extent (BLUESHIFT) **16**

2.6.2.7 Zone tolerance (BLUEFUZZ) **17**

2.6.2.8 Stem width information (STEMWIDTHPROPS) **17**

2.6.2.8.1 Standard horizontal width (STDHW) **17**

2.6.2.8.2 Standard vertical width (STDVW) **18**

2.6.2.8.3 Stem snap horizontal (STEMSNAPH) **18**

2.6.2.8.4 Stem snap vertical (STEMSNAPV) **18**

2.6.2.8.5 Force bold (FORCEBOLD) **19**

2.6.2.8.6 Language group (LANGUAGEGROUP) **19**

2.6.3 Glyph properties (T1GLYPH) **20**

2.6.3.1 Glyph procedure encryption (GLYPHENCRYPT) **20**

2.6.3.2 Glyph prefix length (LENIV) **20**

2.6.3.3 Accent Component Table (ACCENTENCODING) **20**

2.6.3.4 Subroutines (SUBRS) **21**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.6.3.5 Glyph procedure properties (GLYPHPROCPROPS) **21**

2.6.3.5.1 Glyph name property (GNAME) **21**

2.6.3.5.2 Glyph procedure (GLYPHPROC) **21**

2.7 Glyph procedure semantics **22**

2.7.1 Virtual machine **22**

2.7.1.1 Glyph procedure tokens **23**

2.7.1.2 Glyph procedure interpreter **23**

2.7.1.3 Operand list **24**

2.7.1.4 State variables **24**

2.7.1.4.1 Current point **24**

2.7.1.4.2 Escapement **24**

2.7.1.4.3 Current path **24**

2.7.1.4.4 Horizontal stems **24**

2.7.1.4.5 Vertical stems **24**

2.7.1.4.6 Dot section **24**

2.7.1.5 Implicit results stack **25**

2.7.2 Data types **25**

2.7.2.1 Object types **25**

2.7.2.1.1 Integer **25**

2.7.2.1.2 Number **25**

2.7.3 Glyph procedure operators **25**

2.7.3.1 Operators for starting and finishing **26**

2.7.3.1.1 Reference point and escapement (rpe) **27**

2.7.3.1.2 Horizontal reference point and escapement (xrpe) **27**

2.7.3.1.3 End glyph (endglyph) **28**

2.7.3.1.4 Standard indexed accented glyph (siag) **28**

2.7.3.2 Path construction operators **28**

2.7.3.2.1 Closepath (closepath) **29**

2.7.3.2.2 Horizontal lineto (hlineto) **29**

2.7.3.2.3 Horizontal moveto (hmoveto) **29**

2.7.3.2.4 Horizontal-vertical curveto (hvcurveto) **29**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.7.3.2.5 Relative lineto (rlineto) **29**

2.7.3.2.6 Relative moveto (rmoveto) **29**

2.7.3.2.7 Relative relative curveto (rrcurveto) **29**

2.7.3.2.8 Vertical-horizontal curveto (vhcurveto) **30**

2.7.3.2.9 Vertical lineto (vlineto) **30**

2.7.3.2.10 Vertical moveto (vmoveto) **30**

2.7.3.2.11 Set current point (setcurrentpoint) **30**

2.7.3.3 Hint operators **30**

2.7.3.3.1 Dot section (dotsection) **30**

2.7.3.3.2 Horizontal stem (hstem) **31**

2.7.3.3.3 Horizontal stem 3 (hstem3) **31**

2.7.3.3.4 Vertical stem (vstem) **31**

2.7.3.3.5 Vertical stem 3 (vstem3) **32**

2.7.3.4 Arithmetic operator **32**

2.7.3.4.1 Divide (div) **32**

2.7.3.5 Subroutine and subroutine-related operators **32**

2.7.3.5.1 Call subroutine (callsubr) **32**

2.7.3.5.2 Return (return) **33**

2.7.3.5.3 Call utility subroutine (callutilsubr) **33**

2.7.3.5.4 Return value (retval) **33**

2.8 Subroutines **33**

2.8.1 Calling utility subroutines **34**

2.8.1.1 Calling utility subroutine 0 ..... **34**

2.8.1.2 Calling utility subroutine 1 ..... **34**

2.8.1.3 Calling utility subroutine 2 ..... **35**

2.8.1.4 Calling utility subroutine 3 ..... **35**

2.8.2 Hint substitution (Utility subroutine 3) **35**

2.8.3 Flex mechanism **36**

2.8.4 First four Subrs entries **38**

2.8.5 Composite glyphs **39**

2.8.5.1 Using the **siag** operator to create composite glyphs **39**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.8.5.2 Using subroutines to create composite glyphs **39**

2.9 Interchange format **39**

2.9.1 Extensions to font interchange format for type 1 glyph shape information **39**

2.9.1.1 ASN.1 interchange format **40**

2.9.1.2 SGML interchange format **42**

2.9.2 Interchange format for glyph procedures **43**

2.9.2.1 Glyph procedure number representation **43**

2.9.2.2 Glyph operator interchange representation **43**

2.9.2.3 Glyph procedure encryption **43**

Annexes

A Default Accent Component Table **46**

B Compatibility with the installed base **51**

B.1 Compatibility with installed base of glyph procedure interpreters **51**

B.1.1 GlyphEncrypt property (GLYPHENCRYPT) **51**

B.1.2 lenIV property (LENIV) **51**

B.1.3 Hint zone values for ghost stems **51**

B.1.4 Round stem up (ROUNDSTEMUP) **51**

B.1.5 MinFeature (MINFEATURE) **52**

B.1.6 Password (PASSWORD) **52**

B.1.7 Accent Component Table **52**

B.1.8 Interchange format **52**

B.1.8.1 ASN.1 interchange format **52**

B.1.8.2 SGML interchange format **52**

C Composite glyphs **54**

C.1 Composite glyphs **54**

C.1.1 Using the siag operator to create composite glyphs **54**

C.1.2 Using subroutines to create composite glyphs **55**

D Interchange sample **57**

Alphabetical index **59**

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9541-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 18, *Document processing and related communication.*

ISO/IEC 9541 consists of the following parts, under the general title *Information technology — Font information interchange*:

— *Part 1: Architecture*

— *Part 2: Interchange Format*

— *Part 3: Glyph shape representation*

— *Part 4: Application-specific requirements*

Part 1 of ISO/IEC 9541 specifies the architecture of a font resource, i.e., the font description, font metrics, glyph description and glyph metrics properties required for font references and the interchange of font resources.

Part 2 of ISO/IEC 9541 specifies the interchange formats for font information, and the minimum subsets of that information required for interchange.

Part 3 of ISO/IEC 9541 specifies the architecture and interchange formats for glyph shape representations.

Part 4 of ISO/IEC 9541 specifies the architecture and interchange format extensions for application specific (e.g. typesetting of mathematics) requirements.

Annex A forms an integral part of this part of ISO/IEC 9541. Annexes B, C and D are for information only.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Introduction

The use of open networks for the interchange of documents in both office and publishing environments has shown the need for a mechanism enabling the interchange of font information.

It is foreseen that publishing and office technologies will merge and that this development will be facilitated by definition of a standard font resource architecture and a limited number of standard font resource interchange formats.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Information technology — Font information interchange —

Part 3:
Glyph shape representation

Section 1: General

1.1 Scope

ISO/IEC 9541, as a whole, specifies the architecture of font resources, as well as the formats for font interchange among information processing systems. It also specifies the architecture and formats that can be used to construct font references in general electronic document interchange.

This part of ISO/IEC 9541 specifies the architecture and interchange formats of glyph shape representations.

Font resources represented using the architecture and interchange formats defined in parts 1 and 2 of ISO/IEC 9541 are used in various document processing environments in which ASN.1 or SGML parsing algorithms are recognized. The encoding of font resource information as defined in this part of ISO/IEC 9541 is specified in both ASN.1 and SGML representations for consistent generation of font resources for use in these processing environments.

1.2 Conformance

A font resource conforming to this part of ISO/IEC 9541 is a conforming ISO/IEC 9541 font resource. The font resource must conform to the conformance conditions stated in clause 2 of ISO/IEC 9541-2:1991. A conforming implementation of the glyph procedure interpreter shall have the following minimum capabilities:

— represent at least numbers in the range of −8 000 to +8 000 with at least 12 bits of fractional information;

— hold at least 24 objects in the operand list.

1.3 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9541. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9541 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ISO/IEC 8824:1990, *Information technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 8825:1990, *Information technology — Open Systems Interconnection — Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).*

ISO 8879:1986, *Information processing — Text and office systems — Standard Generalized Markup Language (SGML).*

ISO/IEC 9070:1991, *Information technology — SGML support facilities — Registration procedures for public text owner identifiers.*

ISO/IEC 9541-1:1991, *Information technology — Font information interchange — Part 1: Architecture.*

ISO/IEC 9541-2:1991, *Information technology — Font information interchange — Part 2: Interchange Format.*

ISO/IEC 10036:1993, *Information technology — Font information interchange — Procedure for registration of glyph and glyph collection identifiers.*

1.4 Notation

The formal structure of glyph shape properties is specified using the BNF notation described in clause 4 of ISO/IEC 9541.

1.5 Overview of glyph shape representation

Each glyph shape representation technique makes use of different properties in specifying glyph shapes and therefore has its own architecture and interchange format. In this part of ISO/IEC 9541 each glyph shape representation technique is defined in a separate section. The glyph shape representation currently defined is

— ISO Standard Type 1 Glyph Shape Representation (specified in section 2)

NOTE 1 This part of ISO/IEC 9541 may be extended in the future by the addition of further sections specifying additional glyph shape representation techniques.

Glyph shape representations are divided into two broad categories: outline and bitmap representations of glyph shapes.

An outline representation describes a glyph using a mathematical description of the edges of glyph shapes. This has the advantage of allowing transformations such as scaling, rotation, and skewing, and permits many variations of style without additional storage requirements. An outline format also facilitates incorporation of added scaling information, called hints, which aid in the preservation of proportions for all sizes of raster grids (however, their usefulness is not confined to raster devices). Hints can also aid in achieving nonlinear scaling as an optical correction for different absolute sizes of presented glyphs.

For raster devices, outline fonts are converted, after adjustments for scaling requirements, to bitmap representations for final imaging and presentation. However, the presentation of outline glyph shape descriptions is not limited to raster devices; it may also include vector devices such as plotters, signage cutters, engraving machines, or variable spot size raster and gravure devices. Different shape representation techniques may vary in their appropriateness for different presentation devices.

Bitmap representations describe the pattern of pels which are required for printing on raster devices. Bitmap glyph representations are less capable of being scaled or transformed in arbitrary ways while retaining a high standard of typographic quality. Bitmaps of glyph shapes can be represented either as ordered columns or rows of dots, or by a variety of schemes designed to provide more compact representations, particularly for larger sizes.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

1.6 Specification of glyph shapes (GSHAPES)

Any font resource conforming to ISO/IEC 9541 and containing glyph shape information shall contain a GSHAPES property. GSHAPES is a property-list of shape-property-lists defining the sets of shape information associated with this font resource.

```
shapes-property-list ::= shapes-name, shapes-value-property-list
shapes-name ::= STRUCTURED-NAME  -- ISO/IEC 9541-3//GSHAPES
shapes-value-property-list ::= (t1-shape-property-list | property-list)*
```

This architecture allows any glyph shape representation to be defined. The architecture for ISO/IEC 9541 Standard Shape Representation Type 1 is defined in section 2.

1.7 Extensions to the font interchange format

ISO/IEC 9541 font information shall be interchanged using either the ASN.1 or SGML forms defined in ISO/IEC 9541-2. These interchange formats include "markers" to include a definition for interchange formats for glyph shape information. These formats are defined in this clause with further definitions of the detailed format for each glyph shape representation included in equivalent clauses in each of the following sections.

1.7.1 ASN.1

```
ISO9541-GSHAPES { 1 0 9541 3 0 } DEFINITIONS ::= BEGIN
IMPORTS T1-Shape-Property-List  FROM ISO9541-GST1 { 1 0 9541 3 0 0 }
Glyph-Shapes    ::= SET {
t1-shape-property-list     [0]  EXTERNAL  T1-Shape-Property-List  OPTIONAL,
                                -- see Section 2 of this part
non-iso-properties         [99] IMPLICIT Property-List            OPTIONAL }
```

1.7.2 SGML

```
<!-- (c) International Organization for Standardization 1994
     Permission to copy in any form is granted for use with conforming
     SGML systems and applications as defined in ISO 8879:1986;
     provided this notice is included in all copies. -->
<!-- Public document type definition.  Typical invocation:
     <!DOCTYPE gshapes PUBLIC "ISO 9541-3:1994//DTD Glyph Shapes//EN">  -->
<!ELEMENT gshapes - o  (t1shapes? & niprop*)            -- GLYPHSHAPES -->
<!-- Type 1 shape information. Typical invocation:
     <!DOCTYPE t1shapes PUBLIC "ISO 9541-3:1994/DTD Type 1 Glyph Shapes//EN
-->
```

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Section 2: Type 1 glyph shape representation

2.1 Scope

This section specifies the architecture and interchange format of one standard Glyph Shape Representation: ISO/IEC 9541 Standard TYPE 1. This representation technique is appropriate for, but not limited to, presentation on raster devices of low, moderate, and high resolution.

2.2 Definitions

The following definitions are specific to this section.

2.2.1 ciphertext: Information which has been encrypted.

2.2.2 current point: The last point referenced by a path-construction operator in the glyph description language; this point may be either the end point of the most recently drawn line or curve or the point most recently positioned by an rmoveto or setcurrentpoint operator.

2.2.3 encryption key: Integer number required for encrypting or decrypting a glyph procedure.

2.2.4 font program: A computer program incorporating a structured set of glyph procedures and descriptive data.

2.2.5 ghost stem: An imaginary horizontal stem, applied only at the Max-Y and Min-Y extents of a glyph, for which an hstem hint operator must be specified if correct vertical alignment is required.

2.2.6 glyph procedure: A computer program written using the standard glyph shape description operators defined in this section and represented in the interchange format defined herein.

2.2.7 glyph procedure interpreter: A computer process capable of interpreting a glyph procedure for the purpose of constructing an outline representation of the glyph and the associated data structure required for scan conversion.

2.2.8 hint: A procedural or declarative specification of information additional to the geometric shape of a glyph, which aids in the preservation of proportions and features of that glyph during rasterizing for presentation on a raster device.

2.2.9 overshoot: The part of a rounded or pointed glyph extremity which extends slightly beyond the position of flat-shaped extremities; used to achieve optically correct vertical alignment of glyphs for the horizontal writing mode.

2.2.10 path: A possibly disjoint set of subpaths and regions, constructed by one or more path construction operators; the set of all subpaths in a glyph is a single path.

2.2.11 plaintext: Information which is not encrypted.

2.2.12 rasterization: A two-step process which involves determining the areas of a glyph which must be filled, and then creating a bitmap to represent those areas.

2.2.13 region: A geometric area.

2.2.14 segment: The curve or straight line generated by a single graphics drawing command; the curve or straight line between consecutive points on the path defining a glyph shape.

2.2.15 snap: A mechanism for forcing a collection of glyph stems of varying nominal widths (in their device independent outline representation) to convert to the same width when converted to a bitmap representation.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.2.16 subpath: A sequence of connected straight or curved line segments constructed by one or more path construction operators.

2.2.17 typographic color: The relative boldness of the presented font; based on the relative width of the dominant type of stems in the font; also a comparative measure of the surface area of the presented glyph shapes relative to the area of the presentation surface.

2.3 Overview of Type 1 glyph shape representation architecture

Type 1 glyph shape representation is an outline font representation which uses graphical drawing operators to describe the shape of a glyph. This approach has the advantage that the glyph can be scaled, rotated, and transformed in various ways, and can be rasterized to create a bitmap for presentation. Properties defining glyph shapes may be either procedural constructs or additional declarative data defined at the font or glyph level.

The glyph procedures may contain optional declarative hints. Hints consist of information in the font program which assists the glyph procedure interpreter in preserving the proportions and features of a glyph as it is scaled and optimized for the pel grid of a raster device, but their usefulness is not limited to this application. Font-level hints may be defined for the entire font and help to control the alignment and stem widths. Glyph-level hints apply only to the glyph in which they are defined, and primarily help to control the rasterization of vertical and horizontal stems.

Glyph shapes are defined by procedural constructs and additional declarative data called *properties*. These additional data help to accomplish a variety of goals:

— General Properties help to identify the font program and the type of glyph shape representation used (see 2.6.1).

— Typographic color properties help to control the apparent weight of scaled and presented glyphs as well as to achieve uniform stroke widths (see 2.6.2).

— Glyph procedure properties primarily provide parameters and subroutines for use by glyph procedures (see 2.6.3).

NOTE 2 Glyph shape representations which conform to this part of ISO 9541 constitute computer programs or procedures and as such may be subject to protection under laws covering intellectual property rights.

This section defines the properties of glyph shapes in the following clauses:

2.4 explains Type 1 glyph shape concepts
2.5 explains the glyph procedure interpreter model
2.6 defines Type 1 glyph shape properties
2.7 defines glyph-level procedure semantics
2.8 defines the use of subroutine procedures
2.9 defines the interchange format

2.4 Type 1 glyph shape concepts

The following clauses explain concepts related to the Type 1 glyph shape technology.

2.4.1 Glyph coordinate system

The Type 1 glyph shape technology uses a continuous glyph coordinate system as described in subclause 8.2 of ISO/IEC 9541-1:1991.

NOTE 3 Throughout this section, any example value specified as a constant number of units is based on an assumption of a RELUNITS value of 1 000.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.4.2 Glyph procedure language

The glyph procedure language is used to specify glyph shapes and hinting parameters. This language allows the specification of one path, which may consist of multiple disjoint subpaths. After constructing the entire path, the glyph procedure interpreter causes the glyph to be filled or stroked as one entity, depending on the value of the PAINTTYPE property.

2.4.3 Glyph procedure interpreter

The glyph procedure interpreter is a virtual machine (see 2.7.1) which interprets glyph procedures written in the glyph procedure language.

2.4.4 Alignment position

An alignment position is a single position offset which refers to a y-coordinate at which glyph extremities may align in the y direction. Alignment positions are only used in pairs to define alignment zones (see 2.4.8) for font resources with WRMODENAME values of LEFT-TO-RIGHT or RIGHT-TO-LEFT. A font resource with an ALIGNNAME value of BASE-ALIGN has its baseline on the x-axis. Alignment positions are specified by the BLUEVALUES and OTHERBLUES font-level properties and are specified in pairs consisting of a *flat* and an *overshoot* position (see 2.4.5 and 2.4.6).

Font resources with a WRMODENAME value of TOP-TO-BOTTOM, or an ALIGNNAME value of CENTRE-ALIGN, such as those containing East Asian ideographic glyphs, generally have glyphs centered on a single alignment position (which need not be specified in the font), and do not have separate alignment positions for extremities of the glyph shapes.

2.4.5 Flat position

A flat position is the alignment position to which primarily flat shaped glyph extremities align. The judgment as to what a flat shape is depends on an artistic decision by the typeface designer. Extremities which are judged to be almost flat may also be considered to be appropriate to align to a flat position. The concept of a flat position is not applicable to fonts with an ALIGNNAME value of CENTRE-ALIGN, or a WRMODENAME value of TOP-TO-BOTTOM.

2.4.6 Overshoot position

The overshoot position is an alignment position associated with, and occurring just beyond, a flat position (see figure 1), to which non-flat glyph extremities may align. The purpose is to allow glyphs of various shapes to appear to align precisely, allowing for optical illusions, in the vertical direction. The overshoot position is not applicable to fonts with an ALIGNNAME value of CENTRE-ALIGN, or a WRMODENAME value of TOP-TO-BOTTOM.

NOTE 4 Which glyphs require overshoots, and how much a glyph extremity extends past the flat position in the ideal outline representation, is a decision made by the typeface designer or developer. The Outline Modification and Intelligent Fill (rasterization) process (not part of this part of ISO/IEC 9541: see figure 7) may control these extensions so that the resulting bitmap representation exhibits correct vertical alignment for the required size of glyph and characteristics of the presentation device.

2.4.7 Overshoot suppression

Overshoot suppression refers to a mechanism for not allowing overshoots to be rasterized at small sizes even though the scaling process might otherwise cause the overshoot features to round to one pel beyond the flat position.

2.4.8 Alignment zone

An alignment zone is the area lying between two alignment positions—beginning at a flat position and extending to the maximum overshoot position of the tops of glyphs (termed a *top zone*), or from a maximum overshoot

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

position to the flat position of the bottoms of glyphs (a *bottom zone*). There may be multiple such zones. One bottom and two top zones are shown in figure 1. The concept of an alignment zone is not applicable to fonts with an ALIGNNAME value of CENTRE-ALIGN, or a WRMODENAME value of TOP-TO-BOTTOM.

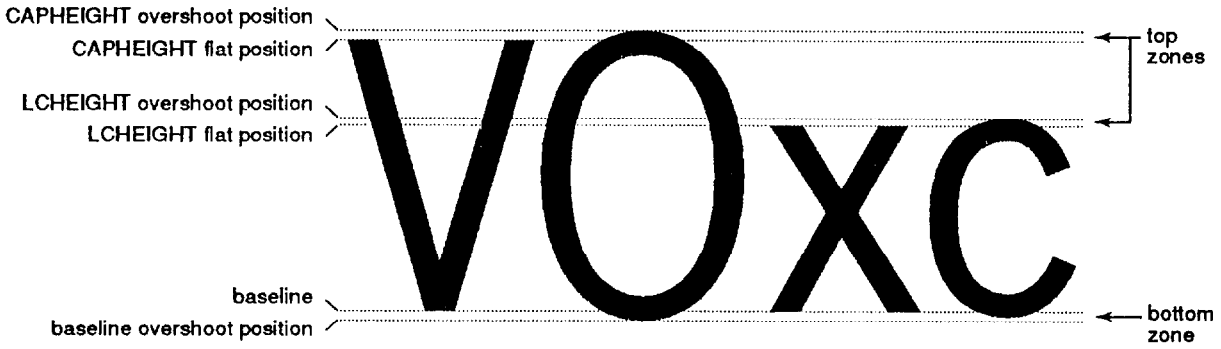


Figure 1 — Three alignment zones, showing flat and overshoot positions for top and bottom zones

2.4.9 Hints

Hint information may be of two general types: font-level and glyph-level. Font-level hints are declared as properties and contain parameters which apply to all glyphs in the font. Font-level hints allow specification of alignment zones to control vertical alignment as well as to control stem widths for all glyphs in the font resource (see 2.6.2.8.1 through 2.6.2.8.5). Glyph-level hints are declared in glyph procedures, using glyph procedure operators, and apply only to the glyph procedure in which they are defined.

Within a glyph procedure, glyph-level declarative hints may be added to control the rasterization of horizontal and vertical stems. Stems may be either straight or curved. The hint operators indicate to the glyph procedure interpreter the *hint zone* (see 2.4.10) which extends from the coordinate specified by the first operand to the relative position indicated by the second operand. The *hstem* hint is used to control the number of pels, measured vertically, to which a horizontal stem is converted. The *vstem* hint is used to control the number of pels, measured horizontally, to which a vertical stem is converted. For the hint operators to be effective, the operands must refer to the exact coordinates of segment endpoints which are at the extremities of the stem.

This allows the glyph procedure interpreter to adjust the glyph outlines to help preserve design features and proportions for scaling to various sizes and resolutions. Figure 2 shows examples of hstem and vstem hints as applied to a Kanji glyph. This illustration shows a detail of the glyph which shows the path segment endpoints and the corresponding boundaries of the associated hstem hint zone.

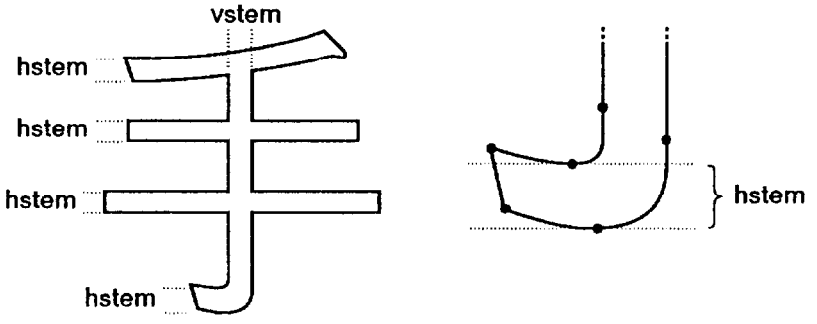


Figure 2 — Hstem and vstem examples

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Figure 3 illustrates the appropriate hint zones for an uppercase "D" glyph, along with showing the hstem and vstem operand value relationships.

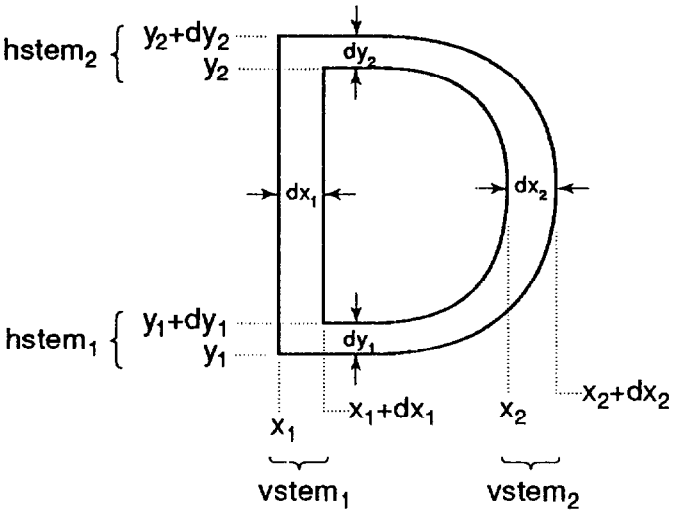


Figure 3 — Glyph level hint zones: the relationship between stems, hint zones, and hint operator operands

When hint zones with the same orientation (for example, two hint zones defined by two vstem hint operators) have overlapping coordinates within a glyph, as shown in figure 4, proportions and features can best be preserved by replacing the originally declared hint zones with a new set at the appropriate place in the glyph drawing procedure. This process of hint substitution is described in 2.8.2.

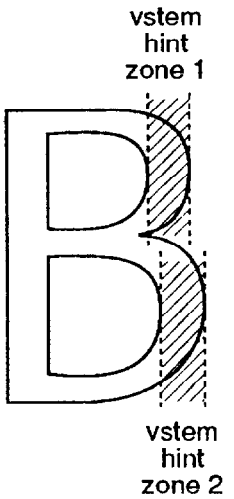


Figure 4 — Overlap zones

A font which requires alignment to multiple alignment positions such as a Latin font (as opposed to a center-aligned script such as Kanji) shall have stem hints specified at, or in, a font-level alignment zone. In some glyphs this happens naturally. For example, figure 5 shows an uppercase "E" where there are three hstem hints declared

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

for the three horizontal stems. For a Latin font, the top stem's hstem hint should extend to the capital height position, and the bottom stem's hstem hint should extend down to the baseline.

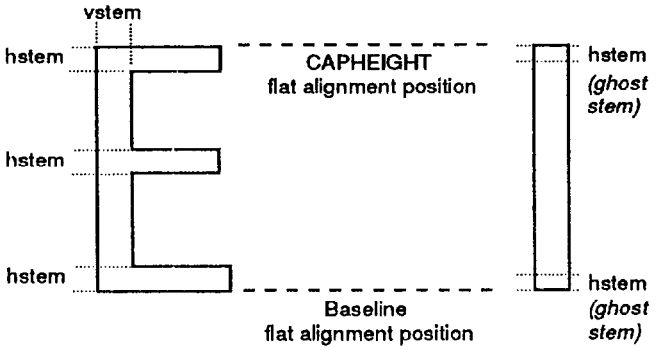


Figure 5 — Hint zones and alignment positions

In a sans serif uppercase "I", however, there are no horizontal stems for these hstem hints. In order to have the capital height and baseline alignments apply to this glyph, it shall have hstem hints (called *ghost* stems, see the Definitions clause) defined for non-existent horizontal stems at these positions. If these hstem hints are not included in the glyph, the results are unspecified.

NOTE 5 For compatibility with the installed base of glyph procedure interpreters, the values allowed for the width of these ghost stems are restricted; see annex B.

If a glyph is to align to font-level alignment positions, its hstem hint may not both have its top at or in a top zone and have its bottom at or in a bottom zone. In the uppercase "I" example, there cannot be just one hstem hint that stretches from baseline to the capital height line; instead one hstem should be at the capital height and another should be at the baseline.

NOTE 6 Several subclauses which discuss hint properties make recommendations for values for these properties. These are not mandatory values, but failure to follow the stated general guidelines may cause unexpected and possibly undesirable results on existing glyph procedure interpreters.

2.4.10 Hint zone

A hint zone is the area between and including the positions defined by a pair of operands to a glyph-level hint operator. The outer boundary specifies the exact location of path segment endpoints at the extremities of glyph features such as stems.

2.4.11 Path direction

A counterclockwise definition of a subpath indicates to the glyph procedure interpreter that the region is to be filled. A clockwise definition of a subpath indicates that the region is contained within a filled region and is to remain unfilled.

2.4.12 Reference point

The reference point is specified at the beginning of a glyph procedure by the operand of an rpe or xrpe operator. Each glyph procedure in a Type 1 font must specify a reference point; all coordinates specified in subsequent path construction or hint operators are interpreted as being relative to this point.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.4.13 Flex mechanism

The Flex mechanism consists of a collection of three utility subroutines intended for use with shallow curves such as those used for cupped serifs. When these features are rasterized at small sizes, the Flex mechanism may be used to substitute a straight line. See 2.8.

2.4.14 Hint substitution

The hint substitution mechanism uses a utility subroutine to change the position of the hint zones for the current glyph. It is used when stems of the same orientation have overlapping coordinates within a single glyph. See 2.8.

2.4.15 Bézier curve

A Bézier curve is derived from a pair of parametric cubic equations:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + X_0 \quad \ldots (2.1)$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + Y_0 \quad \ldots (2.2)$$

The point X_0,Y_0 is the beginning point, and the point X_3,Y_3 is the end point. Associated with these are the control points X_1,Y_1; X_2,Y_2. The Bézier curve is tangent to the line segment from (X_0,Y_0) to (X_1,Y_1) at the point (X_0,Y_0), and to the line segment from (X_2,Y_2) to (X_3,Y_3) at point (X_3,Y_3). This curve will be referred to as the Bézier curve from (X_0,Y_0) to (X_3,Y_3) with control points (X_1,Y_1) and (X_2,Y_2), as illustrated in figure 6.

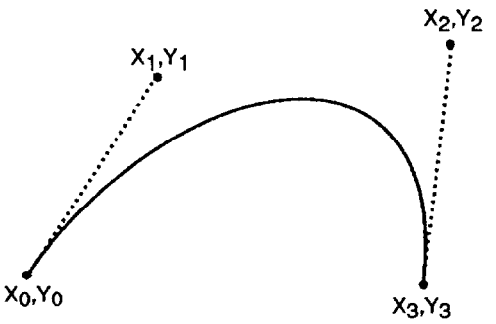


Figure 6 — Bézier curve with control points

The Bézier control points corresponding to this curve are

$$X_1 = X_0 + c_x/3$$

$$Y_1 = Y_0 + c_y/3$$

$$X_2 = X_1 + (c_x + b_x)/3$$

$$Y_2 = Y_1 + (c_y + b_y)/3$$

$$X_3 = X_0 + c_x + b_x + a_x$$

$$Y_3 = Y_0 + c_y + b_y + a_y$$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.5 Glyph procedure interpreter model

Figure 7 shows a model of the glyph procedure interpreter as part of the process of preparing a glyph image for presentation. The glyph procedure interpreter interprets a glyph procedure and constructs the ideal glyph outline and a set of state variables. Only the semantics of this portion of the process are defined by this section. The Outline Modification Algorithm and Intelligent Fill Algorithm use the Geometric Outline, State Variables, and the Font-Level Hints as input to build the appropriate image for presentation. This part of the process is implementation-dependent, and is not defined by this part of ISO/IEC 9541.

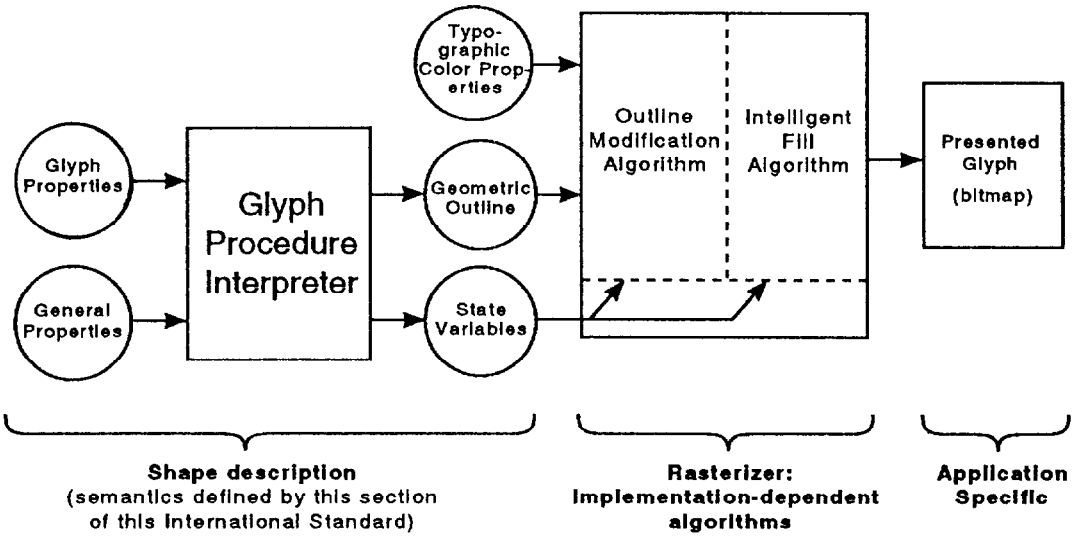


Figure 7 — Type 1 glyph procedure interpreter model

2.6 Type 1 shape properties (T1SHAPES)

T1SHAPES is a property-list of shape properties, defining all the properties for Type 1 shape information. These consist of general font properties, typographic color properties, and glyph properties which have the following formal structure:

```
t1-shape-property-list ::= t1-shape-name,
                           t1-shape-value-property-list
t1-shape-name ::= STRUCTURED-NAME  -- ISO/IEC 9541-3//T1SHAPES
t1-shape-value-property-list ::= ( t1-general-property-list,
                                   t1-color-property-list,
                                   t1-glyph-property-list )
```

2.6.1 General properties (T1GENERAL)

T1GENERAL is a list of global font properties which have the following formal structure:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
t1-general-property-list ::= t1-general-name,
                             t1-general-value-property-list
t1-general-name ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//T1GENERAL
t1-general-value-property-list ::= ( password-property |
                                     painttype-property |
                                        uniqueid-property ) *
```

2.6.1.1 Password (PASSWORD)

The value of the optional PASSWORD property may be included to facilitate use of the font by existing glyph interpreters.

NOTE 7 The PASSWORD property only has significance with respect to compatibility with the installed base. Refer to annex B for the value required for this property for compatibility purposes.

The formal structure of the PASSWORD property is

```
password-property  ::=  password-name,  password-value
password-name  ::=  STRUCTURED-NAME -- ISO/IEC 9541-3//PASSWORD
password-value  ::=  INTEGER
```

2.6.1.2 Paint type (PAINTTYPE)

The value of the mandatory PAINTTYPE property indicates how the glyph contours in the font program are to be rasterized; the value shall be one of the following: 0 = filled; 2 = stroked.

NOTE 8 This property does not indicate how the glyphs will appear when presented, only how the path is treated by the scan conversion process: either filled or stroked. This property is different from the ISO/IEC 9541-1:1991 STRUCTURE property which specifies only whether the final appearance is a solid or outline glyph.

The formal structure of the PAINTTYPE property is

```
painttype-property  ::=  painttype-name, painttype-value
painttype-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//PAINTTYPE
   painttype-value  ::=  INTEGER 0 | 2
```

2.6.1.3 Unique identifier (UNIQUEID)

The value of the optional UNIQUEID may be used to identify the font for the purpose of caching the bitmaps generated from glyph outlines.

NOTE 9 The primary purpose of the UNIQUEID property is to help to uniquely identify bitmaps created and cached from a font program. This is necessary because there is no guarantee that a font's name is unique. A UNIQUEID may allow bitmaps to be cached, either in printer memory or on a hard disk, for all subsequent print jobs instead of only for the current document.

The formal structure of the UNIQUEID property is

```
uniqueid-property  ::=  uniqueid-name, uniqueid-value
uniqueid-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//UNIQUEID
uniqueid-value  ::=  STRUCTURED-NAME
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.6.2 Typographic color properties (T1COLOR)

The typographic color properties may be conceptually divided into two categories: those that define font-level alignment zones and control overshoots, and those that help control stem widths of rasterized glyphs. The alignment and overshoot control properties are generally not applicable to fonts having a value of CENTRE-ALIGN for their ALIGNNAME property, or a WRMODENAME value of TOP-TO-BOTTOM.

The BLUEVALUES property is the only mandatory typographic color property; it shall be included even if the list is empty. Optional properties with no values shall not be included. The stem control category of properties is applicable to any font resource.

The formal structure of the T1COLOR property-list is

```
t1-color-property-list        ::=  t1-color-name,
                                   t1-color-value-property-list
t1-color-name                 ::=  STRUCTURED-NAME
                                   -- ISO/IEC 9541-3//T1COLOR
t1-color-value-property-list  ::=
           ( bluevalues-property | otherblues-property |
             familyblues-poperty | familyotherblues-property |
             bluescale-property | blueshift-property |
             bluefuzz-property | stemwidthprops-property )*
```

NOTE 10 The typographic color properties have names such as "BLUEVALUES" for historical and compatibility reasons. The term *Blue* was originally used in industry as part of a color-coding scheme for the digitization of typefaces. It has proved useful both to group these properties and to link them to the concept of helping to control the typographic color of a font.

2.6.2.1 Top alignment zones (BLUEVALUES)

The value of the mandatory BLUEVALUES property is a value list of pairs of integers which define alignment positions for one bottom zone (the baseline) and up to six top zones. Each integer in each pair of integers represents a vertical position in the glyph coordinate system. The sign of the values is positive in the positive y direction for font resources with WRMODENAME values of LEFT-TO-RIGHT and RIGHT-TO-LEFT; it is not defined for other values. The values shall obey the following rules:

— The first integer in each pair shall be less than or equal to the second integer in that pair.

— The first pair shall be the baseline overshoot and baseline positions.

— Different pairs shall be at least 3 glyph coordinate system units apart from each other and from pairs in the OTHERBLUES list. This minimum distance can be modified by the optional BLUEFUZZ property.

— The maximum difference between values in one pair is constrained as described under the description of the BLUESCALE property.

NOTE 11 Examples of top zone alignment positions which might be defined for Latin scripts may include: LCHEIGHT and LCHEIGHT overshoot position, ascender height and ascender height overshoot position, CAPHEIGHT and CAPHEIGHT overshoot position, figure height and figure height overshoot position, superior flat and superior overshoot position, and ordinal flat and ordinal overshoot position.

An empty list indicates that no alignment zones are necessary for the font. The BLUEVALUES property shall be declared even if the list is empty.

The formal structure of the BLUEVALUES property is

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
bluevalues-property  ::=  bluevalues-name, bluevalues-ordered-value-list
bluevalues-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//BLUEVALUES
bluevalues-ordered-value-list  ::=  (INTEGER,INTEGER)*
                                    -- Maximum of 7 pairs
```

2.6.2.2 Bottom alignment zones (OTHERBLUES)

The value of the optional OTHERBLUES property is a value list of pairs of integers which define alignment positions for up to five bottom zones. Each integer in each pair of integers represents a vertical position in the glyph coordinate system. These bottom zones shall not duplicate the baseline zone required for the BLUEVALUES property. The sign of the values is positive in the positive y direction for font resources with WRMODENAME values of LEFT-TO-RIGHT and RIGHT-TO-LEFT; it is not defined for other values.

The values shall obey the following rules:

— The first integer in each pair shall be less than or equal to the second integer in that pair.

— Different pairs shall be at least 3 glyph coordinate system units apart from each other and from pairs in the BLUEVALUES list. This minimum distance can be modified by the optional BLUEFUZZ property.

— The maximum difference between values in one pair is constrained as described under the description of the BLUESCALE property.

NOTE 12 Typical examples of bottom zone alignment positions for Latin glyphs may include: descender depth overshoot position and descender depth, superior baseline overshoot position and superior baseline, and ordinal baseline overshoot position and ordinal baseline.

The formal structure of the OTHERBLUES property is

```
otherblues-property ::=  otherblues-name,  otherblues-ordered-value-list
otherblues-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//OTHERBLUES
otherblues-ordered-value-list  ::=  (INTEGER,INTEGER)*
                                    -- Maximum of 5 pairs
```

2.6.2.3 Family top alignment (FAMILYBLUES)

The value of the optional FAMILYBLUES property is a list of pairs of integers which define alignment positions which shall be coordinated with those of the standard typeface in a family of typefaces. Each integer in each pair of integers represents a vertical position in the glyph coordinate system. This list specifies the reference values for one bottom zone (the baseline) and up to six top zones, and the contents of this list shall conform to the rules set forth in the list in 2.6.2.1.

The sign of the values is positive in the positive y direction for font resources with WRMODENAME values of LEFT-TO-RIGHT and RIGHT-TO-LEFT; it is not defined for other values.

For family alignment coordination to operate correctly, the following shall be observed with respect to the FAMILYBLUES property:

— One typeface of each family of typefaces is chosen as the standard for that family.

— If all faces in the family are to have their alignments coordinated, the value of the BLUEVALUES property of the standard face shall be copied from the standard face to be the value of the FAMILYBLUES property.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

If the difference between an alignment position in a font and the corresponding family standard alignment position (as defined by FAMILYBLUES) is less than 1 pel, the family standard alignment position shall be used instead of the normal alignment position for that font.

NOTE 13 The FAMILYBLUES property is a means for coordinating the rasterized heights of glyphs from different styles of the same font family. When different styles of the same font family are mixed in text, it is often desirable to coordinate their alignment positions so they will be consistent at small sizes. If these entries are not present, then only a font program's own alignment hints will be considered. Family alignment positions are identical to individual font alignment positions; that is they are items such as the lowercase height and lowercase height overshoot.

The formal structure of the optional FAMILYBLUES property is

```
familyblues-property  ::=  familyblues-name,  familyblues-ordered-value-list
familyblues-name  ::=  STRUCTURED-NAME  --  ISO/IEC 9541-3//FAMILYBLUES
familyblues-ordered-value-list  ::=  (INTEGER,INTEGER)*
                                     -- Maximum of 7 pairs
```

2.6.2.4 Family bottom alignment zones (FAMILYOTHERBLUES)

The value of the optional FAMILYOTHERBLUES property is a value list of pairs of integers which define alignment positions which shall be coordinated with those of the standard typeface in a family of typefaces. Each integer in each pair of integers represents a vertical position in the glyph coordinate system. This list specifies the reference values for up to five bottom zones, and the contents of this list shall conform to the rules set forth in the list in 2.6.2.2.

The sign of the values is positive in the positive y direction for font resources with WRMODENAME values of LEFT-TO-RIGHT and RIGHT-TO-LEFT; it is not defined for other values.

For family alignment coordination to operate correctly, the following shall be observed with respect to the FAMILYOTHERBLUES property:

— One typeface of each family of typefaces is chosen as the standard for that family.

— If all faces in the family are to have their bottom zone alignments coordinated, the value of the OTHERBLUES property of the standard face shall be copied from the standard face to be the value of the FAMILYOTHERBLUES property.

If the difference between an alignment position in a font and the corresponding family standard alignment position (as defined by FAMILYOTHERBLUES) is less than 1 pel, the family standard alignment shall be used instead of the normal alignment for that font.

NOTE 14 The FAMILYOTHERBLUES property is a means for coordinating the rasterized heights of glyphs from different styles of the same font family. When different styles of the same font family are mixed in text, it is often desirable to coordinate their alignment positions so they will be consistent at small sizes. If these entries are not present, then only a font program's own alignment hints will be considered. Family alignment positions are identical to individual font alignment positions; that is they are items such as descender depth, superior baseline, and ordinal baseline.

The formal structure of the FAMILYOTHERBLUES property is

```
familyotherblues-property  ::=  familyotherblues-name,
                                 familyotherblues-ordered-value-list
familyotherblues-name  ::=  STRUCTURED-NAME
                            -- ISO/IEC 9541-3//FAMILYOTHERBLUES
familyotherblues-ordered-value-list  ::=  (INTEGER,INTEGER)*
                                          -- Maximum of 5 pairs
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.6.2.5 Overshoot suppression size (BLUESCALE)

The value of the optional BLUESCALE property specifies the upper limit for the size at which glyph overshoot features shall be rasterized at the same position as the corresponding flat position. This property is not relevant for font resources having an ALIGNNAME property with a value of CENTRE-ALIGN, or for a WRMODENAME value of TOP-TO-BOTTOM.

— For glyph sizes such that RELUNITS glyph coordinate system units occupy fewer device pels than indicated by the BLUESCALE value, overshoot suppression is performed. All features falling in an alignment zone are rasterized at the same pel height.

— For glyph sizes such that RELUNITS glyph coordinate system units occupy the same or greater number of device pels than indicated by the BLUESCALE value, overshoot suppression is turned off, thus allowing overshoots to occur. (This behavior may be modified by the optional BLUESHIFT value; see 2.6.2.6.)

The BLUESCALE value is a number directly related to the number of vertical pels that RELUNITS glyph coordinate space units will occupy before overshoot suppression is turned off. A simple formula that relates this number of pels to the BLUESCALE value is

$$BLUESCALE = pels/RELUNITS \quad \ldots (2.4)$$

NOTE 15 To turn off overshoot suppression at a nominal 46 pels per 1 000 glyph coordinate system units, BLUESCALE should be set to 46/1 000. With this setting of BLUESCALE, overshoot suppression will turn off at proportionately smaller sizes on higher resolution output devices or larger point sizes on lower resolution devices such as screen displays.

The product of the maximum alignment zone height and the pel size for overshoot suppression shall be less than RELUNITS.

NOTE 16 This restriction ensures that overshoot suppression will turn off before the overshoot alignment zone reaches a full device pel. For example, if a font contained a maximum alignment zone height of 23 and RELUNITS = 1 000, then the overshoot suppression size can be 43 pels but not 44.

The formal structure of the BLUESCALE property is

```
bluescale-property  ::=  bluescale-name, bluescale-value
bluescale-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//BLUESCALE
bluescale-value  ::=  REL-RATIONAL
```

2.6.2.6 Overshoot enforcement extent (BLUESHIFT)

The value of the optional BLUESHIFT property is an integer expressed in glyph coordinate units. For glyphs larger than the BLUESCALE size, glyph features that extend beyond the flat position of an alignment zone (above for top-zones, below for bottom-zones) by a glyph coordinate system distance equal to or greater than the value of BLUESHIFT will overshoot, while glyph features closer to the flat position than the BLUESHIFT value will overshoot only if their scan-converted extent is at least one-half pel. If the extent is less than one-half pel, it shall be rasterized to align with the flat position. This property is not relevant for font resources having an ALIGNNAME property with a value of CENTRE-ALIGN, or for a WRMODENAME value of TOP-TO-BOTTOM.

A default value of 7 is assumed if not set explicitly by the BLUESHIFT property. The single setting of BLUESHIFT applies to all alignment zones, regardless of where their overshoot positions lie.

If the Flex mechanism is used, the BLUESHIFT value shall be larger than the maximum Flex feature height. For details, see 2.8.3.

The formal structure of the optional BLUESHIFT property is

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
blueshift-property  ::=  blueshift-name,  blueshift-value
blueshift-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//BLUESHIFT
blueshift-value  ::=  INTEGER
```

2.6.2.7 Zone tolerance (BLUEFUZZ)

The value of the optional BLUEFUZZ property is an integer that specifies the number of glyph coordinate system units to extend (in both directions) the effect of an alignment zone on a horizontal stem. If the top of a horizontal stem is within BLUEFUZZ units outside of a top-zone, the interpreter will act as if the stem top were actually within the zone; the same holds for the bottoms of horizontal stems in bottom-zones. The default value of BLUEFUZZ is 1. This property is not relevant for font resources having an ALIGNNAME property with a value of CENTRE-ALIGN, or for a WRMODENAME value of TOP-TO-BOTTOM.

NOTE 17 BLUEFUZZ is a means for compensating for slightly inaccurate coordinate data. However, the effect of a non-zero value for BLUEFUZZ can usually be better achieved by adjusting the sizes of the alignment zones. If a font's glyphs are precisely aligned, use of this feature is deprecated and the property should be included with the value set to 0.

Because a non-zero value for BLUEFUZZ extends the range of alignment zones, alignment zones should be declared at least (2 * BLUEFUZZ + 1) units apart from each other.

The formal structure of the BLUEFUZZ property is

```
bluefuzz-property  ::=  bluefuzz-name, bluefuzz-value
bluefuzz-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//BLUEFUZZ
bluefuzz-value  ::=  INTEGER
```

2.6.2.8 Stem width information (STEMWIDTHPROPS)

The values of the STEMWIDTHPROPS are a set of font-level stem width hints which enable the glyph procedure interpreter to make stems with slightly different widths to be rasterized the same at small sizes where one pel difference would be very noticeable.

NOTE 18 A common problem with rasterizing glyph shapes is that the nominal stem widths of glyph outlines are not consistent. This can be due to either the intention of the designer or errors in digitization. The following stem width hints can aid the rasterizer in regularizing the widths of stems rasterized for small sizes, while allowing the ideal width to be rasterized when the size and resolution are adequate.

The formal structure of the STEMWIDTHPROPS property is

```
stemwidthprops-property  ::=  stemwidthprops-name, stemwidthprops-property-list
stemwidthprops-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//STEMWIDTHPROPS
stemwidthinfo-property-list  ::=  ( stdhw-property | stdvw-property |
                                    stemsnaph-property | stemsnapv-property |
                                    forcebold-property | languagegroup-property )*
```

2.6.2.8.1 Standard horizontal width (STDHW)

The value of the optional STDHW property is a single-element list containing one rational number entry expressing the width of the dominant horizontal stems (measured vertically in glyph coordinate system units). All horizontal stems with widths within a given tolerance (which is implementation dependent) will be rasterized at the same number of pels as the stem whose width is specified by the value of this operator. If this property is not included, only the glyph-level hints in the current glyph procedure affect how the stems are rasterized.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The formal structure of the STDHW property is

```
stdhw-property  ::=  stdhw-name,  stdhw-value
stdhw-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//STDHW
stdhw-value  ::=  (RATIONAL)+  --  length of list is 1
```

2.6.2.8.2 Standard vertical width (STDVW)

The value of the optional STDVW property is a single-element list containing one rational number entry expressing the dominant width of vertical stems (measured horizontally in glyph coordinate system units). All vertical glyph stems with widths within a given tolerance (specified in pels and supplied by the glyph procedure interpreter) will be rasterized at the same number of pels as the stem whose width is specified by the value of this operator. If this property is not included, only the glyph-level hints in the current glyph procedure affect how the stems are rasterized.

NOTE 19 For best results, the value for this property should be the mean width of the dominant type of stems in the font. For example, in a Latin font, this would be the straight stems of lowercase glyphs. For an italic font program, it will typically be the width of the vertical stem measured at an angle perpendicular to the stem direction.

The formal structure of the STDVW property is

```
stdvw-property  ::=  stdvw-name,  stdvw-value
stdvw-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//STDVW
stdvw-value  ::=  (RATIONAL)+  --  length of list is 1
```

2.6.2.8.3 Stem snap horizontal (STEMSNAPH)

The value of the optional STEMSNAPH property is a list of up to 12 rational numbers of the most common widths (including the dominant width given in the STDHW list) for horizontal stems (measured vertically). These widths shall be given in increasing order. If this property is included, the value of STDHW shall be included. If this property is not included, only glyph-level hints and the value of the STDHW property, if present, shall affect how the stems are rasterized.

This property allows stems of varying widths, but within a glyph interpreter supplied tolerance, to be rasterized with the same pel-width for low resolution devices. If this property is not present, only the glyph-level hints in the current glyph procedure affect how the stems are rasterized.

The formal structure of the STEMSNAPH property is

```
stemsnaph-property  ::=  stemsnaph-name, stemsnaph-value-list
stemsnaph-name  :: =  STRUCTURED-NAME -- ISO/IEC 9541-3//STEMSNAPH
stemsnaph-value-list  ::=  (RATIONAL)+  --  length of list is less
                                            than or equal to 12
```

2.6.2.8.4 Stem snap vertical (STEMSNAPV)

The value of the optional STEMSNAPV property is a list of up to 12 rational numbers of the most common widths (including the dominant width given in the STDVW list) for vertical stems (measured horizontally). These widths shall be given in increasing order. For an italic or oblique font, this list shall be empty or the property may be omitted. If this property is included, the value of STDVW shall be included. If this property is not included, only glyph-level hints and the value of the STDVW property, if present, shall affect how the stems are rasterized.

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

This property allows stems of varying widths, but within a glyph interpreter supplied tolerance, to be rasterized with the same pel-width for low resolution devices. If this property is not present, only the glyph-level hints in the current glyph procedure affect how the stems are rasterized.

The formal structure of the STEMSNAPV property is

```
stemsnapv-property  ::=  stemsnapv-name, stemsnapv-value-list
stemsnapv-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//STEMSNAPV
stemsnapv-value-list  ::=  (RATIONAL)+ --  length of list is less
                                           than or equal to 12
```

2.6.2.8.5 Force bold (FORCEBOLD)

The value of the optional FORCEBOLD property shall be the Boolean value TRUE or FALSE. If a stem of a glyph from a bold font scales to the same number of pels as a stem from a non-bold font from the same family, and the value of FORCEBOLD is TRUE, the scan conversion process may thicken the stem of the glyph from the bold font.

NOTE 20 The optional FORCEBOLD property informs the glyph procedure interpreter that the bold style of a font family must be rasterized to be bolder than the regular font, even at small sizes when the bold and regular might both scale to a one pel-width stem. If a property named FORCEBOLD is included in the typographic color properties, this behavior can be controlled explicitly.

The formal structure of the FORCEBOLD property is

```
forcebold-property  ::=  forcebold-name, forcebold-value
forcebold-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//FORCEBOLD
forcebold-value  ::=  BOOLEAN
```

2.6.2.8.6 Language group (LANGUAGEGROUP)

Certain groups of written languages share broad aesthetic characteristics. Identification of such language groups can prove useful for accurate glyph rasterization.

The value of the optional LANGUAGEGROUP property is an integer that indicates the language group for which the font is primarily intended. If the typographic color property-list does not contain this property, or if the given value is not recognized by the glyph procedure interpreter, then the value of LANGUAGEGROUP defaults to zero. This part of ISO/IEC 9541 currently defines only two language groups, identified as group zero and group one. Additional values for LANGUAGEGROUP are reserved for use by this part of ISO 9541.

Language group 0 consists of scripts such as Latin, Greek, and Cyrillic.

Language group 1 consists of scripts such as Japanese Kanji, Chinese Hanzi, and Korean Hanja.

NOTE 21 The primary characteristic of font resources with a LANGUAGEGROUP value of 1 is that the glyphs may contain significantly more stems than those in a font resource with a value of zero. This may affect decisions by the glyph procedure interpreter with respect to rounding stem widths to the raster grid.

The formal structure of the LANGUAGEGROUP property is

```
languagegroup-property  ::=  languagegroup-name, languagegroup-value
languagegroup-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//LANGUAGEGROUP
languagegroup-value  ::=  INTEGER
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.6.3 Glyph properties (T1GLYPH)

Glyph properties consist of parameters and procedures which are referenced by the glyph procedure interpreter in the process of interpreting glyph procedures. They include a list of component glyphs for making accented glyphs, subroutines which are used for drawing whole or partial glyphs, other hint-related operations, and information about glyph procedure encryption.

The formal structure of the T1GLYPH property-list is

```
t1-glyph-property-list        ::=  t1-glyph-name,
                                   t1-glyph-value-property-list
t1-glyph-name                 ::=  STRUCTURED-NAME
                                   -- ISO/IEC 9541-3//T1GLYPH
t1-glyph-value-property-list  ::=  ( glyphencrypt-property |
                                     lenIV-property |
                                     accentencoding-property |
                                     subrs-property |
                                     glyphprocprops-property-list )*
```

2.6.3.1 Glyph procedure encryption (GLYPHENCRYPT)

The value of the optional GLYPHENCRYPT property is a Boolean which indicates whether the glyph procedures are encrypted or not (see 2.9.2.3 and annex D) or not. A value of TRUE indicates that the glyph procedures are encrypted, a value of FALSE indicates that they are not encrypted. If GLYPHENCRYPT is FALSE, then LENIV shall be included and shall have a value of zero. If the GLYPHENCRYPT property is not specified, its value defaults to TRUE.

The formal structure of the GLYPHENCRYPT property is

```
glyphencrypt-property  ::=  glyphencrypt-name, glyphencrypt-value
glyphencrypt-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//GLYPHENCRYPT
glyphencrypt-value  ::=  BOOLEAN
```

2.6.3.2 Glyph prefix length (LENIV)

The value of the optional property LENIV specifies the number of octets which precede the glyph procedure operators in the glyph procedure interchange format and which are ignored by the glyph procedure interpreter. The default value is 4.

NOTE 22 A value of zero is allowed and will minimize the size of glyph procedures (see annex B).

The formal structure of the LENIV property is

```
leniv-property  ::=  leniv-name, leniv-value
leniv-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//LENIV
leniv-value  ::=  INTEGER
```

2.6.3.3 Accent Component Table (ACCENTENCODING)

A font program may contain an optional property named ACCENTENCODING, which shall be an ordered list of glyph identifiers. If this property is present, then the indexed identifiers of component glyphs referred to by the

--`,,`-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

siag operator (see 2.7.3.1.4) will be interpreted with reference to ACCENTENCODING. If ACCENTENCODING is missing, its value defaults to the list specified in annex A.

If an accented glyph using the **siag** operator tries to reference a glyph code beyond the length of the ACCENTENCODING, the result is not specified.

The formal structure of the ACCENTENCODING property is

```
accentencoding-property  ::=  accentencoding-name, accentencoding-value
accentencoding-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//ACCENTENCODING
accentencoding-value  ::=  (INTEGER, STRUCTURED-NAME)+  --  (see Annex A)
```

NOTE 23 In annex A, columns one and two of the default Accent Encoding Table contain the values required for the integer and structured-name data pair shown in the BNF.

2.6.3.4 Subroutines (SUBRS)

The value of the optional SUBRS property is a list of glyph procedure subroutines. Subroutines may contain any combination of glyph operators including operators to draw paths or subpaths, define hints, or call other subroutines. **Subrs** are accessed by reference to their position in the list, and are subject to the restrictions described in 2.8.

The formal structure of the SUBRS property is

```
subrs-property  ::=  subrs-name, subrs-value-list
subrs-name  ::=  STRUCTURED-NAME -- ISO/IEC 9541-3//SUBRS
subrs-value-list  ::=  (glyphproc)+
glyphproc  ::=  OCTETSTRING  --  see glyph procedure interchange format
```

2.6.3.5 Glyph procedure properties (GLYPHPROCPROPS)

GLYPHPROCPROPS is an ordered property-list of pairs of properties consisting of the glyph name and the glyph shape procedure.

The formal structure of the GLYPHPROCPROPS property is

```
glyphprocprops-property-list  ::=  glyphprocprops-name,
                                   glyphprocprops-value-property-list
glyphprocprops-name  ::=  STRUCTURED-NAME
                          -- ISO/IEC 9541-3//GLYPHPROCPROPS
glyphprocprops-value-property-list  ::=  ( glyphname-property,
                                           glyphproc-property )*
```

2.6.3.5.1 Glyph name property (GNAME)

GNAME is a structured name, and is the name of the glyph for which the shape is described in the following property. This property is defined in ISO/IEC 9541-1.

2.6.3.5.2 Glyph procedure (GLYPHPROC)

GLYPHPROC is an octet string, the procedure for drawing the specified glyph. The glyph procedure shall contain only glyph procedure operators defined by this part of ISO/IEC 9541.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The formal structure of the GLYPHPROC property is

```
glyphproc-property  ::=  glyphproc-name, glyphproc-value
glyphproc-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//GLYPHPROC
glyphproc-value  ::=  OCTET-STRING
```

2.7 Glyph procedure semantics

Unlike the typographic color properties defined in 2.6.2, which are mostly declarative in nature, glyph procedures express the shapes of individual glyphs in the form of a procedural language to be interpreted by the glyph procedure interpreter. This subclause defines a virtual machine which models the glyph procedure interpretation process, and describes the semantics of the glyph procedure language operators and the notation used to describe those semantics.

The virtual machine model is presented only as a model for defining the semantics of glyph procedures. It is not intended to dictate the actual implementation of a glyph procedure interpreter, nor to restrict the method by which glyph procedure semantics are realized in a particular glyph procedure interpreter implementation.

2.7.1 Virtual machine

The process of glyph procedure interpretation is modeled as a virtual machine. The virtual machine model consists of the following components:

— interpreter;
— glyph procedure tokens;
— operand list;
— state variables;
— implicit results stack.

The definitions of the components of the virtual machine depend on four concepts: object, state, operation, and sequence.

Objects are discrete units of data, which can be operated upon by the virtual machine. Each object has a type and a value. The object types defined by this section are specified in 2.7.2.

The *state* of the virtual machine is determined by the combination of the objects in the operand list and a set of data which is private to the virtual machine. This state can be changed by operations performed by the virtual machine.

Operations are actions taken by the virtual machine. These include manipulating objects and manipulating the state of the virtual machine. Operations are affected by operators in glyph procedures, objects used as operands, and the state of the virtual machine prior to the operation.

Sequence is integral to the virtual machine; zero or more tokens are interpreted in a specific order, which may be repeated. Operations performed in sequence have a cumulative effect on the state of the virtual machine.

Figure 8 illustrates a model of the virtual machine.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

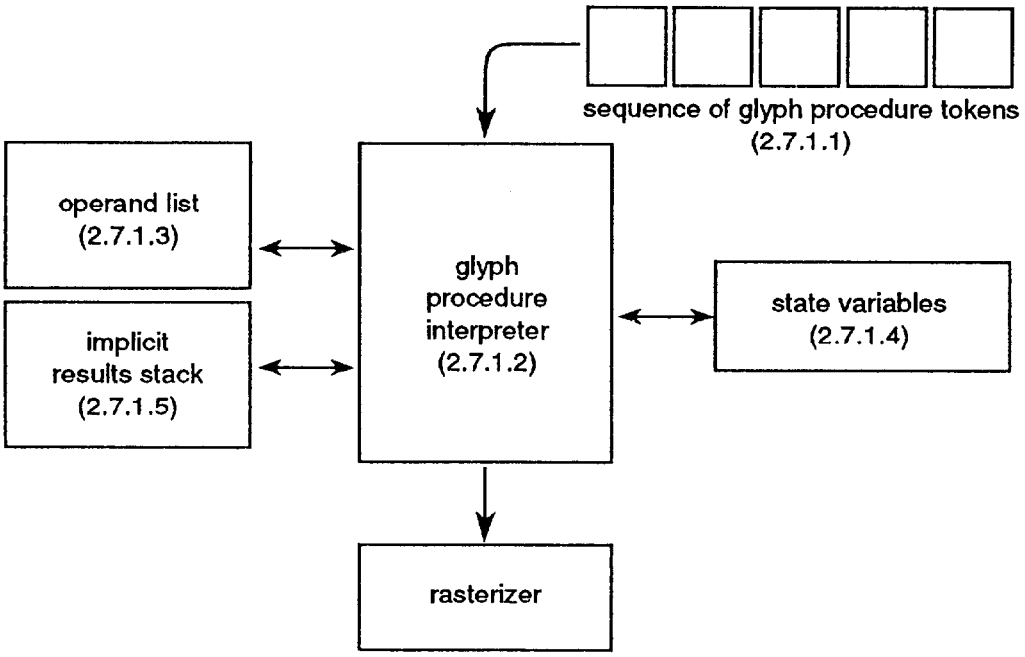


Figure 8 — The virtual machine

The following subclauses specify the components of the virtual machine.

2.7.1.1 Glyph procedure tokens

Each glyph procedure (Octet-string from 2.6.3.5.2) contains a sequence of *tokens* in the Glyph Procedure Interchange Format, as defined in 2.9.2. Each token is either a *literal*, or an *operator*.

A *literal* is a representation, in the Glyph Procedure Interchange Format, of an object. Not all object types have a literal representation in the Interchange Format.

2.7.1.2 Glyph procedure interpreter

The interpreter sequentially interprets the tokens in the glyph procedure. Interpreting a literal token causes the object represented by that literal to be placed onto the head of the operand list.

Interpreting an operator token causes the virtual machine to perform actions depending on the particular operator. The set of operators defined by this section, and their semantics, are defined in 2.7.3.

The actions resulting from interpreting an operator may include:

— remove objects (as operands) from the head or tail of the operand list;
— perform calculations or other manipulations upon objects;
— create new objects;
— place objects (as results) on the head of the operand list;
— clear the operand list;
— change the values of state variables;
— construct or add a path in the glyph coordinate system.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.7.1.3 Operand list

The operand list is a double-ended list which may hold any number of objects (although individual implementations may limit the maximum size of the list). It is a heterogeneous list, in that it may hold any combination of types of objects.

NOTE 24 In this part of ISO/IEC 9541, the terms *list*, *head of list*, *tail of list*, *place*, and *remove* are used in a way consistent with common computer science usage.

Operators requiring objects as explicit operands remove them from the head or the tail of the operand list, and operators which return objects as explicit results place them on the head of the operand list.

The objects on the operand list constitute part of the state of the virtual machine.

2.7.1.4 State variables

A component of the virtual machine is a set of *state variables*. The values of these variables act as implicit operands for many operations, and form part of the state of the virtual machine. The values of all state variables are reset to their default values at the start of interpretation of each glyph procedure.

The following subclauses define each state variable, its type, and its initial value at the start of interpretation of any given glyph procedure.

2.7.1.4.1 Current point

The *CurrentPoint* state variable is a Number pair specifying a point in the Glyph Coordinate System. Its initial value is undefined; its value is established by the required **rpe** or **xrpe** operator which must occur before any path construction operators are executed. The *CurrentPoint* specifies the point in the Glyph Coordinate System relative to which the next path segment will be constructed. The value of the *CurrentPoint* state variable may be modified by several operators including **rmoveto**, **setcurrentpoint**, **rpe**, and **xrpe**.

2.7.1.4.2 Escapement

The vaue of the *Escapement* state variable is a pair of Numbers which specify a point in the Glyph Coordinate System which will become the new *CurrentPoint* after the glyph procedure is executed. Its initial value is undefined; its value is modified by the **rpe** or **xrpe** operator.

2.7.1.4.3 Current path

The value of the *CurrentPath* state variable is the current state of the path being constructed by the glyph procedure. Its initial value is null. The value of the *CurrentPath* state variable is modified by path construction operators.

2.7.1.4.4 Horizontal stems

The value of the *HorizontalStems* state variable is a list of pairs of Numbers defining horizontal hint zones. The initial value of the *HorizontalStems* state variable is null. The value of the *HorizontalStems* state variable may be modified by the **hstem** or **hstem3** operators.

2.7.1.4.5 Vertical stems

The value of the *VerticalStems* state variable is a list of pairs of Numbers defining vertical hint zones. The initial value of the *VerticalStems* state variable is null. The value of the *VerticalStems* state variable may be modified by the **vstem** or **vstem3** operators.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.7.1.4.6 Dot section

The value of the *DotSection* state variable identifies subpaths of the *CurrentPath* which are not to be affected by font-level alignment zones or glyph-level hint zones. The initial value of the *DotSection* state variable is null. Its value is modified by the **DotSection** hint operator.

2.7.1.5 Implicit results stack

The results of calls to **UtilSubrs** are placed on the *implicit results stack*. A **retval** operator may be used to remove one item from the *implicit results stack* and place it on the head of the operand list.

2.7.2 Data types

Objects processed by the virtual machine have an associated object type, and a value. This clause defines the object types used in this International Standard.

2.7.2.1 Object types

2.7.2.1.1 Integer

An object of type Integer is an integer number. The literal representation for objects of type Integer is described in 2.9.2.

2.7.2.1.2 Number

An object of type Number can have a fractional representation.

2.7.3 Glyph procedure operators

The operators which make up the language used to create glyph procedures are divided into five groups by function:

— operators for starting and finishing a glyph's outline;
— path construction operators;
— hint operators;
— arithmetic operators;
— subroutine operators.

The general form of the glyph procedure notation is

operand-list **operator** *result-list*

The encoding value of each operator is shown in parentheses following the operator name. Operator encoding values may consist of either a one-octet or a two-octet value.

NOTE 25 The notation used for the operator encoding value is of the form xx/yy, where xx and yy are both decimal numbers in the range of 00 through 15 (decimal). The most significant nibble is represented by xx, and the least significant nibble by yy. Hence the equivalent decimal number is equal to 16(xx) + yy.

Glyph procedure operators may either take no operands or take their operands from the head or tail of the double-ended operand list. Table 1 shows the symbols used to precede the glyph procedure notation to indicate the action taken.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1 — Symbols preceding glyph procedure notation

operand-list notation	Action
—	takes no operand
op(1) ... op(n)	removes operands from the head of the list
* op(1) ... op(n)	removes operands from the tail of the list

After execution, glyph procedure operators may either leave the operand list unchanged, explicitly clear the list, or place results on the head of the list. Table 2 shows the symbols which follow the glyph procedure notation to indicate the results.

Table 2 — Symbols following glyph procedure notation

results-list notation	Action
—	list unchanged (existing operands on the list may have been referenced, but are not removed)
result(1) ... result(n)	add results to head of list (after removing its operands, if any)
*	list cleared

Because many operators clear the operand list, operands may not, in general, be accumulated on the glyph procedure interpreter operand list for later removal by a sequence of commands. Operands generally may be supplied only for the next operator. Notable exceptions occur with subroutine calls and with the **div** operator.

2.7.3.1 Operators for starting and finishing

A glyph procedure shall begin with either the **rpe** or **xrpe** operator to specify a glyph's *reference point* and *escapement*. Each glyph procedure shall end with an **endglyph** operator, except when using the **siag** operator to build composite glyphs (see 2.7.3.1.4). Figure 9 illustrates a state diagram for the glyph procedure interpreter.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

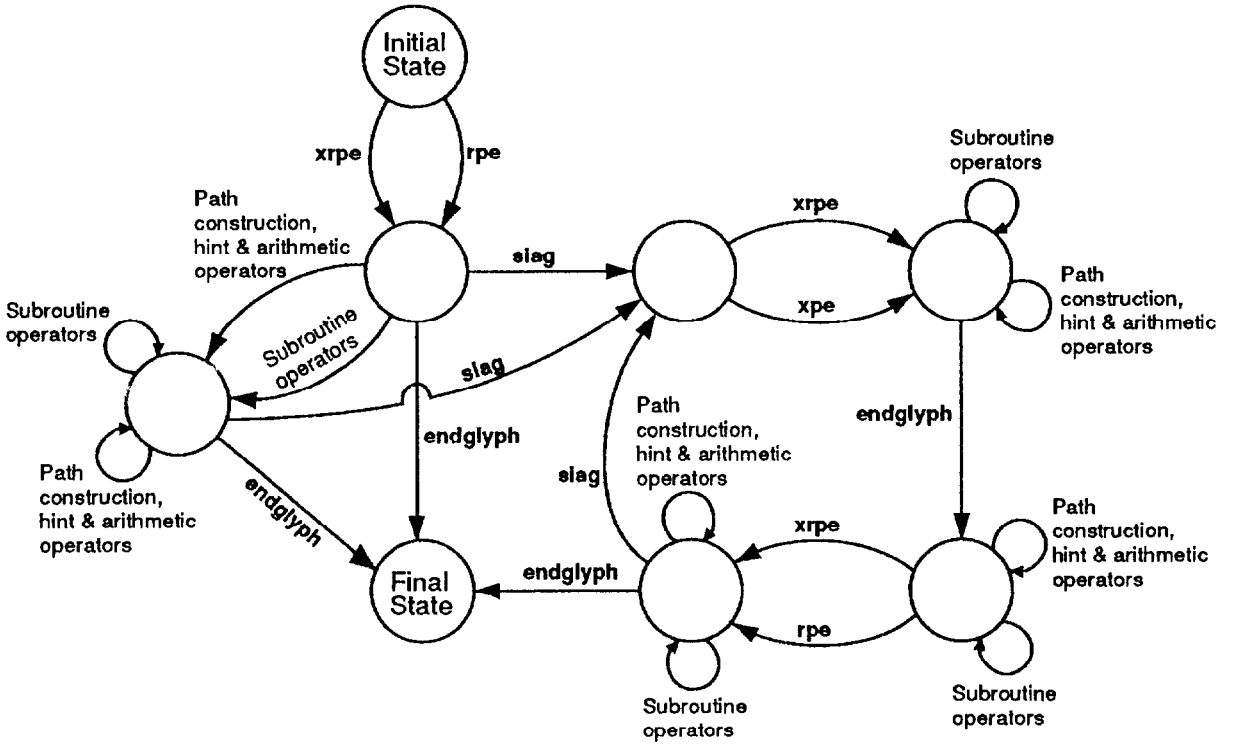


Figure 9 — Glyph procedure interpreter state diagram

2.7.3.1.1 Reference point and escapement (rpe)

```
*  rpx  rpy  ex  ey  rpe  (00/12  00/07)  *
```

specifies the glyph's reference point and escapement. The operands rpx, rpy, ex, and ey are of type Number. The rpx, rpy operands set a reference point at (rpx, rpy) for subsequent relative path construction operators. The **rpe** operator sets the value of the *CurrentPoint* state variable to (rpx, rpy). The *Escapement* state variable is set to (ex, ey). If the values of both rpy and ey are 0, then the **xrpe** operator may be used instead of the **rpe** operator.

The **rpe** operator does not place the reference point in the glyph path. The **rmoveto** operator shall be used for the first point in the path. Either **rpe** or **xrpe** shall be used once, and only once, before any path-construction or hint specification operators are used in a glyph procedure. In non-marking glyphs, such as the space glyph, the reference point shall be (0, 0).

The ex and ey operands duplicate the EX and EY components of the glyph metric information in ISO/IEC 9541-1:1991; in a conforming font these values shall be identical for at least one writing mode of the font.

NOTE 26 The **rpe** operator is useful as a means for reducing storage requirements by reducing the absolute value of some glyph operator operands, since values from −107 through 107 can be expressed in a single octet.

NOTE 27 In fonts with a WRMODENAME value of LEFT-TO-RIGHT, it is recommended that the value of rpx should be equal to the minx value of ISO/IEC 9541-1//EXT, minus ISO/IEC 9541-1//PX. For optimal compatibility with the installed base of glyph interpreters, the value of rpy should be zero.

2.7.3.1.2 Horizontal reference point and escapement (xrpe)

```
*  rpx  ex  xrpe  (00/13)  *
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

specifies the glyph's reference point and escapement. The rpx and ex operands are of type Number. The rpx operand specifies a reference point at (rpx, 0) for subsequent relative path construction operators. The **xrpe** operator sets the value of the *CurrentPoint* state variable to (rpx, rpy). The *Escapement* state variable is set to (ex, ey).

The **xrpe** operator does not place the reference point in the glyph's path. The **rmoveto** operator shall be used for the first point in the path. Either **rpe** or **xrpe** shall be used once, and only once, before any path construction or hint specification operators are used in a glyph procedure. In non-marking glyphs, such as the space glyph, the reference point shall be (0, 0).

The ex operand duplicates the EX component of the glyph metric information in ISO/IEC 9541-1:1991; in a conforming font these values shall be identical for at least one writing mode of the font.

NOTE 28 In fonts with a WRMODENAME value of LEFT-TO-RIGHT, the value of rpx should be the minx value of ISO/IEC 9541-1//EXT, minus ISO/IEC 9541-1//PX.

2.7.3.1.3 End glyph (endglyph)

– **endglyph** (00/14) *

finishes a glyph outline definition and shall be the last operator in a glyph's procedure. The **endglyph** causes the glyph procedure interpreter to paint the glyph's path in the manner indicated by the PAINTTYPE property. If the **endglyph** operator does not terminate a glyph procedure, the glyph cannot be rasterized. It shall not be used for composite glyphs defined using the **siag** operator, since each component glyph procedure shall be terminated with an **endglyph** operator.

2.7.3.1.4 Standard indexed accented glyph (siag)

* arp adx ady bglyph aglyph **siag** (00/12 00/06) *

This operator constructs an accented glyph by indexed reference to two component glyphs in the same font. The arp, adx, and ady operands are of type Number; bglyph and aglyph are of type Integer and are non-negative. The arp operand is the x component of the reference point of the accent; this value shall be the same as the rpx operand to the **xrpe** or **rpe** operator in the accent's own glyph procedure. The composite glyph is constructed by placing the reference point of the accent at an offset of (adx, ady) relative to the reference point of the base glyph. The bglyph operand is an indexed reference to the glyph identifier of the base glyph, and the aglyph operand is an indexed reference to the glyph identifier of the accent glyph. Both bglyph and aglyph are indexes into the Accent Component Table. See C.1.1 for an explanation of using the **siag** operator for constructing composite glyphs.

If the glyph identifiers of both components of an accented glyph do not appear in the Accent Component Table, the accented glyph cannot be built using the **siag** operator. The component glyphs shall not themselves be composite glyphs created with the **siag** operator.

The **rpe** or **xrpe** operator that begins the composite glyph shall be the same operator and have the same operands as the corresponding operator in the base glyph (the glyph referenced by the bglyph operand).

The **siag** operator is the last operator in the glyph procedure for the accented glyph because the accent and base glyph's procedures each already end with their own **endglyph** operator.

NOTE 29 The use of this operator saves space in a Type 1 font program, but its use is restricted to those glyphs whose components are defined in the Accent Component Table. The recommended general method for constructing multi-component composite glyphs is by the use of subroutines (Subrs). See annex C for more information.

2.7.3.2 Path construction operators

The following operators may be used to construct a path for the current glyph. A subpath should not cross itself or intersect other subpaths; if it does, the results are implementation-specific. A subpath which is to be filled shall be defined in the counterclockwise direction. A subpath which is not to be filled shall be defined in the clockwise direction.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.7.3.2.1 Closepath (closepath)

– **closepath** (00/09) *

closepath closes a subpath.

NOTE 30 All glyph subpaths should end with a **closepath** operator, otherwise if the path is stroked (by setting PAINTTYPE to 2), unexpected behavior may result. This operator does not reposition the current point. Any subsequent **rmoveto** must be relative to the current point in force before the **closepath** operator was given.

2.7.3.2.2 Horizontal lineto (hlineto)

* dx **hlineto** (00/06) *

appends a horizontal line segment, of length dx, to the *CurrentPath*. The operand dx is of type Number.

2.7.3.2.3 Horizontal moveto (hmoveto)

* dx **hmoveto** (01/06) *

repositions the *CurrentPoint* by adding dx units in the horizontal direction. The operand dx is of type Number. Equivalent to dx 0 **rmoveto**.

2.7.3.2.4 Horizontal-vertical curveto (hvcurveto)

* dx1 dx2 dy2 dy3 **hvcurveto** (01/15) *

appends a Bézier cubic curve segment to the *CurrentPath*; it is equivalent to dx1 0 dx2 dy2 0 dy3 **rrcurveto** (See 2.7.3.2.7). The operands dx1, dx2, dy2, and dy3 are of type Number.

NOTE 31 This operator eliminates two operands from an **rrcurveto** call when the first Bézier tangent is horizontal and the second Bézier tangent is vertical.

2.7.3.2.5 Relative lineto (rlineto)

* dx dy **rlineto** (00/05) *

appends a straight line segment to the *CurrentPath*; the number pair is interpreted as a displacement relative to the *CurrentPoint*. The operands dx and dy are of type Number.

2.7.3.2.6 Relative moveto (rmoveto)

* dx dy **rmoveto** (01/05) *

starts a new subpath of the *CurrentPath* by setting the *CurrentPoint* to (x+dx, y+dy) where (x, y) was the previous *CurrentPoint*, without adding any segments to the *CurrentPath*. The operands dx and dy are of type Number.

2.7.3.2.7 Relative relative curveto (rrcurveto)

* dx1 dy1 dx2 dy2 dx3 dy3 **rrcurveto** (00/08) *

adds a Bézier cubic curve segment to the *CurrentPath* between the *CurrentPoint*, shown in figure 6 as (X_0,Y_0), and the point (X_3,Y_3), using (X_1,Y_1) and (X_2,Y_2) as the Bézier cubic control points (see definition in 2.4.15). The operands for **rrcurveto** are all of type Number and are expressed as relative to the previous point. Hence the operands are derived as follows:

$dx1 = X_1 - X_0$

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

$dx2 = X_2 - X_1$

$dx3 = X_3 - X_2$

$dy1 = Y_1 - Y_0$

$dy2 = Y_2 - Y_1$

$dy3 = Y_3 - Y_2$

After constructing the curve, **rrcurveto** makes X_3,Y_3 the new *CurrentPoint*.

2.7.3.2.8 Vertical-horizontal curveto (vhcurveto)

```
*  dy1  dx2  dy2  dx3  vhcurveto  (01/14)  *
```

for vertical-horizontal curveto, this operator appends a curve segment to the *CurrentPath*; it is equivalent to 0 dy1 dx2 dy2 dx3 0 **rrcurveto** (see 2.7.3.2.7). The operands dy1, dx2, dy2, and dx3 are of type Number.

NOTE 32 This operator eliminates two operands from an **rrcurveto** call when the first Bézier tangent is vertical and the second Bézier tangent is horizontal.

2.7.3.2.9 Vertical lineto (vlineto)

```
*  dy  vlineto  (00/07)  *
```

appends a vertical line segment, of length dy, to the *CurrentPath*. Equivalent to 0 dy **rlineto**.

2.7.3.2.10 Vertical moveto (vmoveto)

```
*  dy  vmoveto  (00/04)  *
```

repositions the *CurrentPoint* by adding dy units in the vertical direction. Equivalent to 0 dy **rmoveto**. dy is of type Number; it may be positive (increasing y) or negative (decreasing y).

2.7.3.2.11 Set current point (setcurrentpoint)

```
*  x  y  setcurrentpoint  (00/12  02/01)  *
```

sets the *CurrentPoint* in the glyph procedure interpreter to (x, y) in absolute glyph coordinate system coordinates without executing an **rmoveto** operator. This establishes the current point for a subsequent relative path construction operator. The **setcurrentpoint** operator shall be used only in conjunction with results from Utility Subroutine 0.

2.7.3.3 Hint operators

The use of hint operators is optional in glyph procedures.

2.7.3.3.1 Dot section (dotsection)

```
–  dotsection  (00/12  00/00)  *
```

delimits a subpath. This operator indicates that the delimited subpath shall not be controlled by any font-level hint zones in which it may lie. The **DotSection** operator modifies the *DotSection* state variable.

If the **dotsection** hint operator is used, it shall be used in pairs. The first **dotsection** operator shall occur immediately after the first **rmoveto** that begins the delimited subpath of the glyph. The second **dotsection** operator shall be given immediately after the **closepath** that finishes the subpath.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NOTE 33 When a subpath lies in or near a font-level alignment or glyph-level hint zone, but does not align exactly with that zone, the subpath may be distorted by the scan conversion process so that it does align with that zone. If the hint replacement mechanism is not available, or is available but not used, the resulting adjustments to subpath positions may be considered to be a distortion for some glyphs at some sizes. Hint replacement is the preferred method to achieve the most accurate rasterization of these glyph features. However, for compatibility with glyph procedure interpreters not capable of performing hint replacement, the **dotsection** operator may be employed in addition to hint replacement. Hence, this operator may be used for the dots in glyphs such as "i", "j", and "!", but may also be used for any subpath which can benefit from its use.

2.7.3.3.2 Horizontal stem (hstem)

```
* y dy hstem (00/01) *
```

places a pair of Numbers into the *HorizontalStems* state variable list. The operands y and dy are of type Number, and define a horizontal stem hint zone between the y coordinates y and y±dy, where y is relative to the y coordinate of the reference point. The y and dy values shall correspond to the position of either the points on a straight stem, or the extreme points of a curved stem, or the results are not specified.

NOTE 34 If horizontal stem zones within a single glyph overlap each other, sub-optimal results may occur. Hint replacement may be used to avoid stem hint overlaps. For more details on hint replacement, see 2.8.2.

2.7.3.3.3 Horizontal stem 3 (hstem3)

```
* y0 dy0 y1 dy1 y2 dy2 hstem3 (00/12 00/02) *
```

places three pairs of operands, of type Number, into the *HorizontalStems* state variable list. These Numbers define three horizontal stem hint zones between the y coordinates y0 and y0±dy0, y1 and y1±dy1, and between y2 and y2±dy2; where y0, y1 and y2 are all relative to the y coordinate of the reference point.

The **hstem3** operator sorts these zones by the y values to obtain the lowest, middle and highest zones, called ymin, ymid and ymax respectively. The corresponding dy values are called dymin, dymid and dymax. These stems and the counters between them will all be controlled. These coordinates shall obey certain restrictions:

— dymin = dymax

— The distance from ymin+dymin/2 to ymid+dymid/2 shall equal the distance from ymid+dymid/2 to ymax+dymax/2. In other words, the distance from the center of the bottom stem to the center of the middle stem shall be the same as the distance from the center of the middle stem to the center of the top stem.

If a glyph procedure uses an **hstem3** operator in the hints for a glyph, the glyph procedure shall not use **hstem** commands. If hint replacement is performed, either **hstem** or **hstem3** (or neither) shall be solely used within a single glyph procedure and its subroutines.

NOTE 35 The **hstem3** operator is especially suited for controlling the stems and counters of symbols with three horizontally oriented features with equal vertical widths and with equal white space between these features, such as the mathematical equivalence symbol or the division symbol.

2.7.3.3.4 Vertical stem (vstem)

```
* x dx vstem (00/03) *
```

places a pair of Numbers into the *VerticalStems* state variable list. The operands x and dx are of type Number, and define a vertical stem hint zone between the x coordinates x and x±dx, where x is relative to the x coordinate of the reference point. The operands x and dx are of type Number. The x and dx values shall correspond to the position of either the points on a straight stem, or the extreme points of a curved stem, or the results are not specified.

NOTE 36 If vertical stem zones within a single glyph overlap each other, sub-optimal results may occur. Hint replacement may be used to avoid stem hint overlaps. For details on hint replacement, see 2.8.2.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.7.3.3.5 Vertical stem 3 (vstem3)

```
* x0 dx0 x1 dx1 x2 dx2 vstem3 (00/12 00/01) *
```

places three pairs of operands, of type Number, into the *VerticalStems* state variable list. These Numbers define three vertical stem hint zones between the x coordinates x0 and x0±dx0, x1 and x1±dx1, and x2 and x2±dx2, where x0, x1 and x2 are all relative to the x coordinate of the reference point.

The **vstem3** operator sorts these zones by the x values to obtain the leftmost, middle and rightmost zones, called xmin, xmid and xmax respectively. The corresponding dx values are called dxmin, dxmid and dxmax. These stems and the counters between them will all be controlled. These coordinates shall obey certain restrictions described as follows:

— dxmin = dxmax

— The distance from xmin+dxmin/2 to xmid+dxmid/2 shall equal the distance from xmid+dxmid/2 to xmax+dxmax/2. In other words, the distance from the center of the left stem to the center of the middle stem must be the same as the distance from the center of the middle stem to the center of the right stem.

If a glyph procedure uses a **vstem3** operator in the hints for a glyph, the glyph procedure shall not use **vstem** commands. If hint replacement is performed, either **vstem** or **vstem3** (or neither) shall be solely used within a single glyph procedure and its subroutines.

NOTE 37 The **vstem3** operator is especially suited for controlling the stems and interior white spaces of glyphs such as a lowercase "m".

2.7.3.4 Arithmetic operator

2.7.3.4.1 Divide (div)

```
num1 num2 div (00/12 00/12) quotient
```

divides num1 by num2. The operand num1 is pushed on the head of the operand list, followed by num2. After execution, the result is pushed on the head of the operand list. The operands num1 and num2 are of type Number, and the quotient is of type Number.

2.7.3.5 Subroutine and subroutine-related operators

2.7.3.5.1 Call subroutine (callsubr)

```
subr_index callsubr (00/10) result
```

calls the subroutine in the subr_index position in the Subrs list in the GLYPHPROC properties list. The operand subr_index is of type Integer. Each element of the Subrs list is identical in format to a glyph procedure and in the same interchange format.

Operands placed on the head of the glyph procedure interpreter operand list prior to calling the subroutine, and the zero or more results placed on the head of the list by the subroutine, are according to the subroutine definition. Although **callsubr** does not explicitly leave any results on the operand list, operators in the subroutine may leave results on the list.

Subroutines are generally used to represent sequences of path construction or hint operators which are repeated throughout the font program, or for calling Utility Subroutines. Subroutine calls shall not be nested more than 10 deep. See 2.8.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.7.3.5.2 Return (return)

– **return** (00/11) –

returns from a subroutine in the Subrs list (which has been called with a **callsubr** operator) and continues execution in the calling glyph procedure. Each glyph procedure subroutine named in the Subrs list shall end with a **return** operator.

2.7.3.5.3 Call utility subroutine (callutilsubr)

op1 . . . opn n u **callutilsubr** (00/12 01/00) result

The following sequence shall be used for calling a Utility Subroutine: The *n* operands *op1* through *opn* are provided to the Utility Subroutine. They are removed from the head of the operand list and passed to the u^{th} Utility Subroutine which is then executed. The **retval** operator is used to retrieve results from the Utility Subroutine and place it on the head of the operand list (see 2.7.3.5.4); thus the **callutilsubr** operator may be followed by one or more **retval** operators. See 2.8 for details on using **callutilsubr**.

Utility Subroutines are defined only by this section of this part of this International Standard. Their uses are specified in 2.8.1, and the syntax of the four currently defined Utility Subroutines is defined in 2.8.1.

2.7.3.5.4 Return value (retval)

– **retval** (00/12 01/01) number

causes a returned value to be retrieved from the implicit results stack and placed on the head of the operand list. If used, it shall only be used immediately after the **callutilsubr** operator, and there shall be no intervening operators.

2.8 Subroutines

A font program may make use of two sets of subroutines, Subrs and the standardized Utility Subroutines. Uses for subroutines include:

— Reducing the storage requirements of a font program by combining the statements that describe common elements of the glyphs in the font.

— If the flex feature is required, Subrs 0 through 2 shall be used for that purpose (if the Flex feature is not required, Subrs 0 through 2 may be used for other purposes).

— Subr 3 is reserved for use with the Hint Substitution feature (if Hint Substitution is not required, Subr 3 may be used for other purposes).

— Subrs 4 and higher are available for use with other glyph procedure calls.

— Utility Subroutines 0 through 2 implement the Flex feature.

— Utility Subroutine 3 implements hint replacement.

The Subrs list contains sections of glyph procedures expressed in the proper interchange representation and encrypted as glyph procedures containing glyph shape operators (see 2.9.2). When a font contains repeated elements, these elements may be candidates for being expressed as glyph procedure subroutines.

NOTE 38 The glyph procedure operator set includes only relative forms of path building operators. For example, **rmoveto** and **rlineto** are included, but moveto and lineto are not. Using relative operators facilitates the reuse of subroutines for sections of glyph outlines, regardless of their absolute placement within the glyph.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

An element of the Subrs list is a glyph procedure, and it must end with the **return** operator. These subroutines are called via the **callsubr** operator, using the index in the Subrs list as the operand. Glyph procedure subroutines may call other subroutines, to a depth of 10 nested calls.

The use of glyph procedure subroutines is not a requirement of a Type 1 font program. However, optimized use of subroutines can contribute greatly to reducing storage space.

Utility Subroutines may be used for Hint Replacement and the Flex mechanism. These Utility Subroutine procedures work by using some coordinated Subrs entries as well. Hence, the semantics of the first four Utility Subroutine entries and the first four Subrs entries have fixed meanings. If Hint Replacement or Flex is not used in a font resource, the corresponding Subrs entries may be used for other purposes.

Utility Subroutine entries beyond the first four are reserved for future extensions.

2.8.1 Calling utility subroutines

2.8.1.1 Calling utility subroutine 0

Calling sequence:

```
3  0  callutilsubr  retval  retval  (08/15  08/11  00/12  01/00
                                     00/12  01/01  00/12  01/01)
```

Operands: tolerance, x, y

Returned values: x, y

Semantics: Utility Subroutine 0 is used for the Flex mechanism and is the last call made to finalize a sequence describing a pair of curve segments with Flex. This Utility Subroutine removes three operands from the head of the operand list and returns two results. The first of the three input operands is the tolerance parameter: the size (in hundredths of a device unit) of the rasterized Flex height at which the two curves will be expressed as curves rather than as a straight line. The second and third operands are the coordinates of the end point expressed in absolute terms in glyph coordinate system units (relative to the glyph coordinate system origin).

NOTE 39 The Flex mechanism is used to preserve proportions of features which are too subtle to otherwise rasterize in a pleasing manner at small sizes. For cupped serifs or other features that interact with overshoot zones, 50 (or one-half of a device pel) should be used for this tolerance control parameter. Thus, if the Flex height rasterizes to 50 hundredths of a pel or more, the curves will be used; if less, a straight line will be used. Detailed semantics are defined in 2.8.3.

This Utility Subroutine returns two results, to the implicit results stack, which are the absolute x and y coordinates of the end point of the curve. Two **retval** operators are required to return these values from the implicit results stack to the glyph procedure interpreter's operand list. The two **retval** operators shall be immediately followed by a **setcurrentpoint** operator (see 2.7.3.2.11), which uses the two **return**ed values as operands for updating the *CurrentPoint*.

2.8.1.2 Calling utility subroutine 1

Calling sequence:

```
0  1  callutilsubr  (08/11  08/12  00/12  01/00)
```

Operands: (none)

Returned values: (none)

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Semantics: Utility Subroutine 1 is used for invoking the Flex mechanism, and is the first call made to initiate its operation. This Utility Subroutine requires no operands and returns no results. It saves the coordinates of the *CurrentPoint* at the beginning of the flex curve and initiates the process of accumulating coordinates which are passed to the Flex mechanism by subsequent calls to Utility Subroutine 2. Detailed semantics are defined in 2.8.3.

2.8.1.3 Calling utility subroutine 2

Calling sequence:

```
0  2  callutilsubr  (08/11  08/13  00/12  01/00)
```

Operands: (none)

Returned values: (none)

Semantics: Utility Subroutine 2 is used for the Flex mechanism. This Utility Subroutine requires no operands and returns no results. Each call to Utility Subroutine 2 is preceded by an **rmoveto** operator with its appropriate operands. This causes the coordinates of the point referenced in the **rmoveto** operator to be accumulated for later processing by Utility Subroutine 0. Detailed semantics are defined in 2.8.3.

2.8.1.4 Calling utility subroutine 3

Calling sequence:

```
1  3  callutilsubr  retval  (08/12  08/14  00/12  01/00  00/12  01/01)
```

Operands: subr_index

Returned values: subr_index or 3

Semantics: Utility Subroutine 3 is used for invoking the Hint Substitution mechanism. This Utility Subroutine removes an operand from the head of the operand list, which must be the index of a Subr containing replacement hint operators. If the glyph procedure interpreter is capable of doing hint replacement, the same subr_index will be returned, otherwise the number 3 will be returned. The number 3 is the index of a Subr which shall contain only a **return** operator. A call to Utility Subroutine 3 shall be followed by a **callsubr** operator. Semantics of hint replacement are defined in 2.8.2.

2.8.2 Hint substitution (Utility subroutine 3)

The stem hints, **vstem** and **hstem**, affect the treatment of all subsequent coordinates in a glyph procedure. Occasionally, a glyph outline may require certain stem hints for some part of its outline, but different stem hints for other parts of its outline. After executing the coordinate operators for the current set of stem hints, these hints may be discarded and new stem hints specified at the appropriate place in the sequence of path construction operators.

To discard old stem hints and insert new ones, the new stem hints shall be placed in a glyph procedure subroutine in the Subrs list. This subroutine may be placed at any index in the Subrs list except 0 through 3. Call this subroutine index *subr_index*. This subroutine shall contain only stem hint operators and their operands. Then, at the point in the glyph outline where the old hints are to be discarded and the new ones inserted, insert the following glyph procedure sequence:

```
subr_index 1 3  callutilsubr retval callsubr
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

This sequence of code operates as follows. The Utility Subroutine 3 is called with one operand, the entry in the Subrs list that contains the new hint operators. The *subr_index* is transferred to the subroutine, and the hint substitution procedure is executed. Since not all interpreters will be capable of discarding hints in mid-outline, the hint substitution procedure checks if the interpreter is capable of performing this action. If so, it returns subr_index. If not, it returns the number 3. The **retval** operator transfers the result (either 3 or *subr_index*) from the implicit results stack to the head of the operand list. Finally, the subroutine referenced by the *subr_index* is called by the **callsubr** operator.

Entry 3 in the Subrs list must be a glyph procedure consisting solely of a **return** operator. If the glyph procedure interpreter is not capable of discarding old hints in mid-outline, then this mechanism ignores the new hints.

2.8.3 Flex mechanism

Very shallow curves that are nearly horizontal or nearly vertical in orientation are especially difficult to approximate on a raster device. Examples of such curves include cupped serifs and tapered stems. These features can be controlled with the Flex mechanism, which uses Utility Subroutine entries 0, 1, and 2.

The Flex mechanism determines whether the two Bézier curves which make up the subtle curve should be used as defined, or whether a straight line segment between the two outer endpoints should be used instead. The method calculates whether the height of the Flex feature in device space is less than a height control parameter. If so, then the two curves are replaced by a single straight line segment. If not, the curve points are adjusted so that the curve features will be rasterized appropriately.

A particular curve sequence is a candidate for the Flex mechanism only if the arrangement of points on that curve meets certain conditions (examples are shown in figure 10):

— The sequence must be formed by exactly two Bézier curve segments.

— The outer endpoints must be at the same x (or y) coordinate; in other words, they must be precisely vertical or horizontal.

— The joining endpoint between the two curves and the control points associated with this endpoint must all be positioned at the horizontal (or vertical) extreme of the double curve section. The joining point need not be equidistant from the endpoints of the double curve section.

— The difference in x (or y) coordinates between an outer endpoint and the center (joining) endpoint (the flex height) must be 20 units or less.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

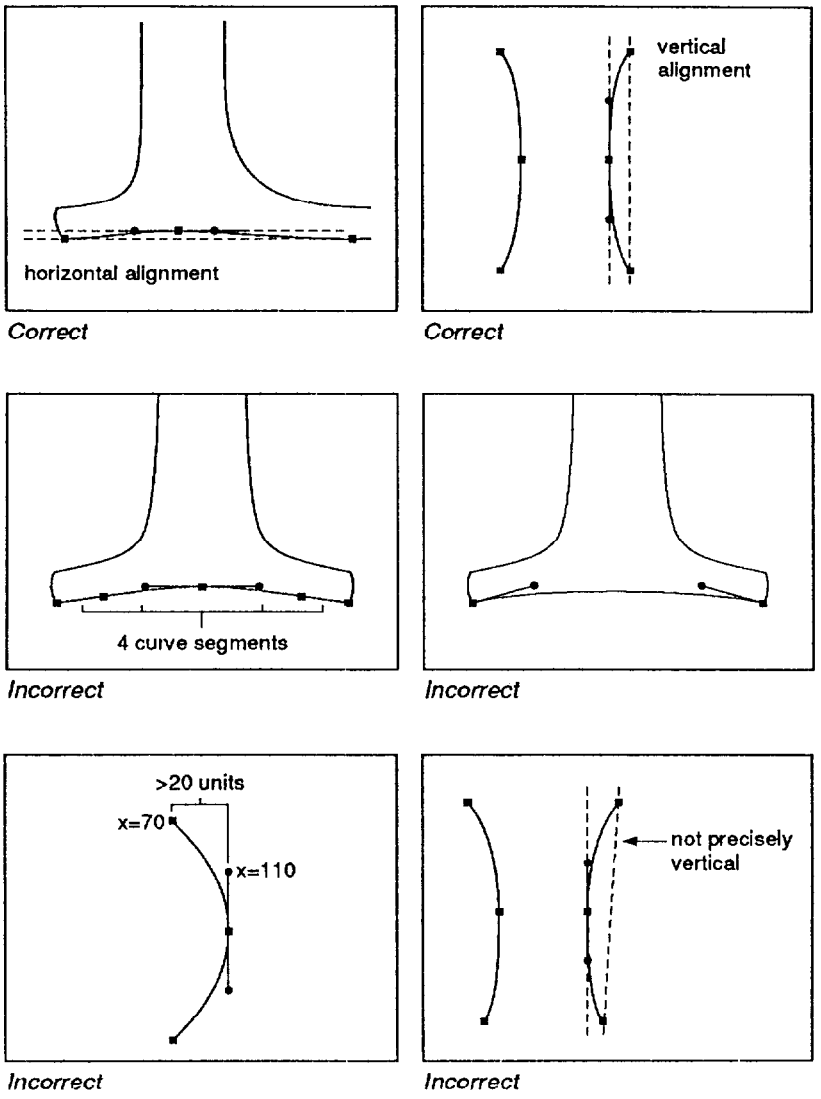


Figure 10 — Appropriate and inappropriate curves for the Flex mechanism

For best results with cupped serifs (and any other shallow curves that lie within an alignment zone), the joining (center) endpoint should be positioned precisely at the flat position of the alignment zone.

Flex features shall be coordinated with the BLUESHIFT property. The BLUESHIFT value shall be larger than the maximum Flex feature height.

NOTE 40 If the maximum Flex feature height is 6 or less, the BLUESHIFT property may be omitted. Since the default value of BLUESHIFT is 7, this property must be set explicitly if the maximum Flex feature height is more than 6.

To add the Flex mechanism to two suitable Bézier curve segments, several changes must be made in the glyph procedure. The following is an algorithm that accomplishes these changes.

a) Note the coordinates of the current point where the first curve begins. Call this the starting point.

b) Compute the relative distance from the starting point to a reference point. For horizontally oriented curves, the reference point will have the same x coordinate as the joining point and the same y coordinate as the starting point. For vertically oriented curves, the reference point will have the same y coordinate as the joining point and the same x coordinate as the starting point.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) Remove the two **rrcurveto** operators, leaving six coordinate values (12 numbers).

d) Recompute the coordinates of the first pair to be relative to the reference point.

e) Insert at the beginning of the sequence the coordinate of the reference point relative to the starting point. There are now seven coordinate values (14 numbers) in the sequence.

f) Place a call to Subrs entry 1 at the beginning of this sequence of coordinates, and place an **rmoveto** operator and a call to Subrs entry 2 after each of the seven coordinate pairs in the sequence.

g) Place the Flex height control parameter and the final coordinate expressed in absolute terms followed by a call to Subrs entry 0 at the end. The height control parameter is an integer expressing hundredths of a pel.

NOTE 41 For cupped serifs, the recommended Flex control parameter is 50 (fifty hundredths of a pel).

2.8.4 First four Subrs entries

If any of the Utility Subroutines are called by the glyph procedures of a font program, the first four entries in the Subrs list in the GlyphProc properties shall be assigned glyph procedures that correspond to the following operator sequences. If neither Flex nor hint substitution is used in the font program, then this requirement is removed, and the first Subrs entries may be normal glyph procedure subroutine sequences.

If the Flex mechanism is referenced by the font program, then the first three Subrs entries must contain:

Subrs index number 0:

```
3  0  callutilsubr  retval  retval  setcurrentpoint  return
```

Subrs index number 1:

```
0 1  callutilsubr  return
```

Subrs index number 2:

```
0  2  callutilsubr  return
```

If the Hint Substitution mechanism is referenced by the font program, then the fourth Subr (index number 3) must contain:

```
return
```

The Subrs entries 1 and 2 are merely abbreviations for calling Utility Subroutines. This saves two glyph procedure octets on each call. Subrs entry 3 is used when the glyph procedure interpreter cannot perform hint replacement. Subrs entry 0 passes the final three operands in the Flex mechanism into Utility Subroutine 0.

Utility Subroutine 0 takes as operands the final coordinates and the Flex parameter, and returns the coordinates of the final point of the flex curve. Subrs entry 0 then transfers these two coordinate values to the head of the operand list with two **retval** operators and uses them as operands to the **setcurrentpoint** operator.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

2.8.5 Composite glyphs

There are two methods for creating composite glyphs in a Type 1 font program:

— Using the **siag** operator (which only permits the combination of two component glyphs per invocation)

— Using Subrs subroutines

The following subclauses describe the requirements for each method, and annex C illustrates the use of each method.

2.8.5.1 Using the siag operator to create composite glyphs

In order to use the **siag** operator, four conditions shall be met:

— A glyph identifier must exist for the composite glyph.

— Glyph identifiers must exist for the individual component glyphs.

— Glyph metrics must exist for the composite and component glyphs.

— The component glyphs must be mapped to glyph indices in the Accent Component Table.

The **siag** operator may then be used to link the two component glyphs into a composite glyph (see C.1.1 for more information on using **siag** for constructing composite glyphs).

2.8.5.2 Using subroutines to create composite glyphs

In order to use subroutines to create composite glyphs, three conditions shall be met:

— A glyph identifier must exist for the composite glyph.

— Glyph metrics must exist for the composite glyph.

— The component sub-glyphs must be represented by glyph procedures in the Subrs list.

The **callsubr** operator may then be used to link two or more component Subrs into a composite glyph (see C.1.2 for more information on using subroutines for constructing composite glyphs).

2.9 Interchange format

2.9.1 Extensions to font interchange format for type 1 glyph shape information

ISO/IEC 9541 font resources shall be defined using either the ASN.1 or SGML format defined in ISO/IEC 9541-2, in conjunction with the ASN.1 or SGML format for glyph shapes defined in section 1. The interchange formats defined below provide the ASN.1 and SGML for type 1 glyph shapes.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.9.1.1 ASN.1 interchange format

```
ISO9541-GST1 { 1 0 9541 3 0 0 } DEFINITIONS ::= BEGIN
IMPORTS Structured-Name  FROM ISO9541-SN { 1 0 9541 2 3 }
T1-Shape-Property-List    ::= SEQUENCE {
  t1-name-prefixes              [0] IMPLICIT SET OF T1-Name-Prefix   OPTIONAL,
  t1-general-props              [1] IMPLICIT T1-General-Property-List,
  t1-color-props                [2] IMPLICIT T1-Color-Property-List,
  t1-glyph-props                [3] IMPLICIT T1-Glyph-Property-List }
T1-Name-Prefix              ::= SEQUENCE {
            -- see global name note at the end of this clause
                                [0]: IMPLICIT Code,
                                [1]: IMPLICIT Structured-Name  }
T1-General-Property-List    ::= SET  {
  password                      [0] IMPLICIT INTEGER              OPTIONAL,
  painttyp                      [1] IMPLICIT Paint-Code                   ,
  uniqueid                      [2] IMPLICIT Global-Name          OPTIONAL }
T1-Color-Property-List      ::= SET  {
  blue-values                   [0] IMPLICIT Top-Zone-List                ,
  other-blues                   [1] IMPLICIT Bottom-Zone-List     OPTIONAL,
  family-blues                  [2] IMPLICIT F-Top-Zone-List      OPTIONAL,
  family-other-blues            [3] IMPLICIT F-Bottom-Zone-List   OPTIONAL,
  blue-scale                    [4] IMPLICIT Rational             OPTIONAL,
  blue-shift                    [5] IMPLICIT INTEGER              OPTIONAL,
  blue-fuzz                     [6] IMPLICIT INTEGER              DEFAULT 1,
  stem-width                    [7] IMPLICIT Stem-Width           OPTIONAL }
T1-Glyph-Property-List      ::= SET  {
  glyph-encrypt                 [0] IMPLICIT BOOLEAN             DEFAULT TRUE,
  lenIV                         [1] IMPLICIT Cardinal             OPTIONAL,
  accent-encoding               [2] IMPLICIT Acc-Enc-List         OPTIONAL,
  subrs                         [3] IMPLICIT Subr-Procedures-List OPTIONAL,
  glyph-procs                   [4] IMPLICIT Glyph-Procedure-Prop-List
                                                                  OPTIONAL }
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
Paint-Code                  ::= ENUMERATED  { filled    (0),
                                              outlined  (2)  }
Top-Zone-List               ::= SEQUENCE OF (INTEGER,INTEGER)
                                            --   length of list <= 7 pairs
Bottom-Zone-List            ::= SEQUENCE OF (INTEGER,INTEGER)
                                            --   length of list <= 5 pairs
F-Top-Zone-List             ::= SEQUENCE OF (INTEGER,INTEGER)
                                            --   length of list <= 7 pairs
F-Bottom-Zone-List          ::= SEQUENCE OF (INTEGER,INTEGER)
                                            --   length of list <= 5 pairs
Stem-Width                  ::= SET  {
  std-hw                         [0] IMPLICIT Std-HW-List          OPTIONAL,
  std-vw                         [1] IMPLICIT Std-VW-List          OPTIONAL,
  stem-snap-h                    [2] IMPLICIT Stem-Snap-H-List     OPTIONAL,
  stem-snap-v                    [3] IMPLICIT Stem-Snap-V-List     OPTIONAL,
  force-bold                     [4] IMPLICIT BOOLEAN              OPTIONAL,
  language-group                 [5] IMPLICIT Lang-Group-Val       OPTIONAL }
Acc-Enc-List                ::= SEQUENCE OF Acc-Enc
Acc-Enc                     ::= SEQUENCE  {
  acc-comp-index                 [0] IMPLICIT INTEGER,
  glyph-name                     [1] IMPLICIT Global-Name }
Subr-Procedures-List        ::= SEQUENCE OF Glyph-Procedure
Glyph-Procedure-Prop-List   ::= SET OF SEQUENCE  {
  glyph-name                     [0] IMPLICIT Global-Name,
  glyph-procedure                [1] IMPLICIT Glyph-Procedure  }
Std-HW-List                 ::= SEQUENCE OF Rational
                                            --   length of list = 1
Std-VW-List                 ::= SEQUENCE OF Rational
                                            --   length of list = 1
Stem-Snap-H-List            ::= SEQUENCE OF Rational
                                            --   length of list <= 12
Stem-Snap-V-List            ::= SEQUENCE OF Rational
                                            --   length of list <= 12
Lang-Group-Val              ::= ENUMERATED { latin-based  (0),
                                             ideographic  (1) }  DEFAULT 0


Glyph-Procedure             ::= OCTET STRING
Global-Name                 ::= SEQUENCE {
                  -- see global name note at the end of this clause
 prefix-index                      [0] IMPLICIT Code                OPTIONAL,
 name-value                        [1] IMPLICIT Structured-Name                }
Code                        ::= INTEGER (1 .. 255)
Rational                    ::= SEQUENCE {
  numerator                    [0] IMPLICIT INTEGER
  denominator                  [1] IMPLICIT INTEGER (1..2147483647) DEFAULT 1 }
```

NOTE 42 The Global-Name and Name-Prefix properties represent an encoding efficiency that permits the use of short Structured-Names (as short as one Object-Name-Component) within the body of a font resource or font reference. The Name-Prefixes property is an indexed list of Structured-Name values (see annex B of ISO/IEC 9541-2 for the definition of Structured-Names) that are to be pre-pended to the Structured-Name value of any Global-Name containing a corresponding prefix-index value. If no prefix-index value is provided in a Global-Name, then the Structured-Name value specified is a fully qualified Structured-Name. Caution should be exercised in the creation of Name-Prefixes since no validation checking can be performed to verify the fully qualified Structured-Name resulting from the combination.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2.9.1.2 SGML interchange format

```
<!-- (C)  International Organization for Standardization 1992 Permission to copy
            in any form is granted for use with conforming SGML systems and
            applications as defined in ISO 8879:1986; provided this notice is
            included in all copies.  -->
<!-- Public document type definition.  Typical invocation:
    <!DOCTYPE t1shapes PUBLIC "ISO/IEC 9541-3:1994//DTD Type 1 Glyph Shapes//EN"   -->
<!ELEMENT  t1shapes  - -  (t1namtbl?, t1genprp, t1colprp, t1gpprp) >


<!ELEMENT  t1namtabl  - o  (prefix , strucnm) +              -- Name Prefix Table
                                  see global name note at the end of this clause -->
<!ELEMENT  prefix     - o  (code)                            -- Name Prefix Index -->
<!ELEMENT  t1genprp   - o  (password? & painttyp & uniqueid?) -- General Prop List -->
<!ELEMENT  password   - o  (int)                                    -- Password -->
<!ELEMENT  painttyp   - o  (int)                                  -- PaintType -->
<!ELEMENT  uniqueid   - o  (glbname)                               -- UniqueID -->
<!ELEMENT  t1colprp   - o  (bluevals  & othrblue? & famblue?  & famoblue? &
                              bluescal? & blueshft? & bluefuzz? & stemwdth? )
                                                  -- Typographic color props -->
<!ELEMENT  (bluevals | famblue)  - o ((int,int)*)              -- BlueValues,
                                                                  FamilyBlues,
                                                                 Maximum of 7 -->
<!ELEMENT  (othrblue | famoblue) - o ((int,int)*)           -- OtherBlues,
                                                              FamilyOtherBlues,
                                                                 Maximum of 5 -->
<!ELEMENT  bluescal   - o  (relr)                                  -- BlueScale -->
<!ELEMENT  blueshft   - o  (int)                                   -- BlueShift -->
<!ELEMENT  bluefuzz   - o  (int)                                    -- BlueFuzz -->
<!ELEMENT  stemwdth   - o  (stdhw?    & stdvw?   & stemsnph? & stemsnpv? &
                            forcebld? & langgrp?)          -- Stem Width Properties -->
<!ELEMENT  (stdhw | stdvw)  - o  (ratl)     -- Standard Horizontal/Vertical Widths -->
<!ELEMENT  (stemsnph | stemsnpv)  - o  (ratl*)    -- Horizontal/Vertical Stem Snap -->
<!ELEMENT  forcebld   - o  (bool)                                  -- ForceBold -->
<!ELEMENT  langgrp    - o  (card)                               -- LanguageGroup -->
<!ELEMENT  t1gpprp    - o  (glncrpt?  & leniv?   & accenlst? & subrs?    &
                            glplists?)                    -- GlyphProc Properties -->
<!ELEMENT  glncrpt    - o  (bool)                          -- glyphencrypt Property -->
<!ELEMENT  leniv      - o  (card)                                    -- lenIV -->
<!ELEMENT  accenlst   - o  (accentpr*)                     -- AccentEncoding -->
<!ELEMENT  accentpr   - o  (accindx, glyphid)      -- AccentComponentIndex/GlyphID -->
<!ELEMENT  accindx    - o  (int)                        -- Accent Component Index -->
<!ELEMENT  subrs      - o  (glyphprc+)                          -- Subroutines -->
<!ELEMENT  glplist    - o  (glprocpr*)                  -- Glyph Procedures List -->
<!ELEMENT  glprocpr   - o  (glyphid, glyphprc)      --GlyphID/Glyph Procedure Pair -->
<!ELEMENT  glyphid    - o  (glbname)              -- GlyphID, see ISO/IEC9541-1:1991-->
<!ELEMENT  glyphprc   - o  (#PCDATA)                       -- Glyph Procedure -->
<!ELEMENT  glbname    o o  (prefix? , strucnm) +           -- Global Name
                                  see global name note at the end of this clause -->
```

NOTE 43 The glbname and nametbl elements represent an encoding efficiency that permits the use of short Structured-Names (as short as one Object-Name-Component) within the body of a font resource or font reference. The nametbl element is an indexed list of Structured-Name values (see annex B of part two of this International Standard for the definition of Structured-Names) that are to be pre-pended to the Structured-Name value of any glbname element containing a corresponding prefix value. If no prefix value is provided in a glbname element, then the strucnm value specified is a fully qualified

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Structured-Name. Caution should be exercised in the creation of namtbl values since no validation checking can be performed to verify the fully qualified Structured-Name resulting from the combination.

2.9.2 Interchange format for glyph procedures

2.9.2.1 Glyph procedure number representation

A glyph procedure octet containing the values from 32 through 255 inclusive indicates an integer. These values are interpreted in four ranges.

a) A glyph procedure octet containing a value, v, between 32 and 246 inclusive, represents the integer (v – 139). Thus, the integer values from –107 through 107 inclusive may be expressed in a single octet.

b) A glyph procedure octet containing a value, v, between 247 and 250 inclusive, indicates an integer involving the next octet, w, according to the formula:

$$[(v - 247) \times 256] + w + 108 \quad \ldots (2.6)$$

Thus, the integer values between 108 and 1131 inclusive can be expressed in 2 octets in this manner.

c) A glyph procedure octet containing a value, v, between 251 and 254 inclusive, indicates an integer involving the next octet, w, according to the formula:

$$-[(v - 251) \times 256] - w - 108 \quad \ldots (2.7)$$

Thus, the integer values between –1131 and –108 inclusive can be expressed in 2 octets in this manner.

d) Finally, if the glyph procedure octet contains the value 255 (15/15), the next four octets are interpreted as the components of a 32-bit two's-complement signed integer with the most significant octet first. Thus, any 32-bit signed integer may be expressed in 5 octets in this manner (the 255 octet plus 4 more octets).

2.9.2.2 Glyph operator interchange representation

Glyph operators are expressed in 1 or 2 octets. Single octet operators are represented in 1 octet that contains a value between 0 (00/00) and 11 (00/11) or between 13 (00/13) and 31 (01/15). Not all possible operator values are defined. The operator values that are omitted are reserved for future use of this International Standard. If an operator octet contains the value 12 (00/12), then the value in the next octet indicates an operator.

NOTE 44 This escape mechanism allows more than thirty-two operators to be expressed in this manner. These 2-octet operators are not used as often as the 1-octet operators; this technique helps to minimize the length of glyph procedures.

2.9.2.3 Glyph procedure encryption

Glyph procedures are encrypted as a barrier to casual inspection and illegal copying of copyrighted font outlines. It is not necessary to allow other encryption methods or encryption keys because the purpose of the font standard is to allow font interchange.

Glyph procedure encryption is a combination of three techniques:

— A pseudo-random number generator generates a sequence of keys that are combined with the plaintext to produce the ciphertext.

— Cipher feedback is employed in the generation of these keys; in other words, each octet of ciphertext is used in the production of the next key.

— Each plaintext sequence has a semi-random sequence of octets inserted at the beginning, so that repeated encryption of the same plaintext will produce different ciphertexts.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

The following algorithms for encryption and decryption are nearly identical, but they differ subtly because it is always the ciphertext octets that are used to generate the next key. It is necessary to use two separate procedures to handle encryption and decryption.

To encrypt a sequence of plaintext octets to produce a sequence of ciphertext octets:

a) Generate n random octets to be additional plaintext octets at the beginning of the sequence of plaintext octets, for some value of n.

b) Initialize an unsigned 16-bit integer variable R to the encryption key.

c) For each octet, P, of plaintext (beginning with the newly added random octets) execute the following steps:

   1) Assign the high order 8 bits of R to a temporary variable, T.

   2) Exclusive-OR P with T, producing a ciphertext octet, C.

   3) Compute the next value of R by the formula ((C + R) * c1 + c2) mod 65536, where c1 is 52845 (decimal) and c2 is 22719 (decimal).

The initial value of the encryption key R shall be 4330.

This encryption step can be performed by the following C language program, where r is initialized to the encryption key:

```
unsigned short int r;
unsigned short int c1 = 52845;
unsigned short int c2 = 22719;

unsigned char Encrypt(plain) unsigned char plain;
{unsigned char cipher;
cipher = (plain ^ (r>>8));
r = (cipher + r) * c1 + c2;
return cipher;
}
```

NOTE 45 For information, the following algorithm is provided. It may be used to decrypt a sequence of ciphertext octets to produce the original sequence of plaintext octets:

a) Initialize an unsigned 16-bit integer variable R to the encryption key (the same key as used to encrypt).

b) For each octet, C, of ciphertext the following steps are executed:

   1) Assign the high order 8 bits of R to a temporary variable, T.

   2) Exclusive-OR C with T, producing a plaintext octet, P.

   3) Compute the next value of R by the formula ((C + R) * c1 + c2) mod 65536, where c1 and c2 are the same constants that were used to encrypt.

c) Discard the first n octets of plaintext; these are the random octets added during encryption. The remainder of the plaintext octets are the original sequence.

The decryption step can be performed by the following C language program, where r is initialized to the key for the encryption type:

```
unsigned short int r;
unsigned short int c1= 52845;
unsigned short int c2 = 22719;

unsigned char Decrypt(cipher) unsigned char cipher;
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
{unsigned char plain;
plain = (cipher ^ (r>>8));
r = (cipher + r) * c1 + c2;
return plain;
}
```

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A
(normative)

Default Accent Component Table

The *Registration Authority Identifier* column lists the registered identifier assigned by the ISO/IEC 10036 Registration Authority.

The *Index* and *Registration Authority Identifier* columns represent the values used for the ACCENTENCODING property's *integer/structured-name* pairs. The *Name* column is informative and lists the name used in the installed base of glyph procedure interpreters. All ordinal entries which are defined with an identifier value of −1 indicate a lack of assignment of a glyph to that index. All numbers are in decimal notation.

NOTE 46 The *Registration Authority Identifier* is a glyph structured name, registered in accordance with ISO/IEC 10036. The identifier shown is the last Object-Name-Component of a structured name having the canonical form: "ISO/IEC 10036/RA/Glyphs::nnnnnn"; where nnnnnn is the identifier shown.

Table A.1

Index	Registration Authority Indentifier	Name
0	−1	–
1	−1	–
2	−1	–
3	−1	–
4	−1	–
5	−1	–
6	−1	–
7	−1	–
8	−1	–
9	−1	–
10	−1	–
11	−1	–
12	−1	–
13	−1	–
14	−1	–
15	−1	–
16	−1	–
17	−1	–
18	−1	–
19	−1	–

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Index	Registration Authority Indentifier	Name
20	–1	–
21	–1	–
22	–1	–
23	–1	–
24	–1	–
25	–1	–
26	–1	–
27	–1	–
28	–1	–
29	–1	–
30	–1	–
31	–1	–
32	32	space
33	33	exclam
34	34	quotedbl
35	35	numbersign
36	164	dollar
37	37	percent
38	38	ampersand
39	39	quoteright
40	40	parenleft
41	41	parenright
42	42	asterisk
43	43	plus
44	44	comma
45	45	hyphen
46	46	period
47	47	slash
48	48	zero
49	49	one

Index	Registration Authority Indentifier	Name
50	50	two
51	51	three
52	52	four
53	53	five
54	54	six
55	55	seven
56	56	eight
57	57	nine
58	58	colon
59	59	semicolon
60	60	less
61	61	equal
62	62	greater
63	63	question
64	64	at
65	65	A
66	66	B
67	67	C
68	68	D
69	69	E
70	70	F
71	71	G
72	72	H
73	73	I
74	74	J
75	75	K
76	76	L
77	77	M
78	78	N
79	79	O

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Index	Registration Authority Indentifier	Name
80	80	P
81	81	Q
82	82	R
83	83	S
84	84	T
85	85	U
86	86	V
87	87	W
88	88	X
89	89	Y
90	90	Z
91	91	bracketleft
92	92	backslash
93	93	bracketright
94	94	asciicircum
95	95	underscore
96	96	quoteleft
97	97	a
98	98	b
99	99	c
100	100	d
101	101	e
102	102	f
103	103	g
104	104	h
105	105	i
106	106	j
107	107	k
108	108	l
109	109	m

Index	Registration Authority Indentifier	Name
110	110	n
111	111	o
112	112	p
113	113	q
114	114	r
115	115	s
116	116	t
117	117	u
118	118	v
119	119	w
120	120	x
121	121	y
122	122	z
123	123	braceleft
124	124	bar
125	125	braceright
126	126	asciitilde
127	–1	–
128	–1	–
129	–1	–
130	–1	–
131	–1	–
132	–1	–
133	–1	–
134	–1	–
135	–1	–
136	–1	–
137	–1	–
138	–1	–
139	–1	–

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Index	Registration Authority Indentifier	Name
140	–1	–
141	–1	–
142	–1	–
143	–1	–
144	–1	–
145	–1	–
146	–1	–
147	–1	–
148	–1	–
149	–1	–
150	–1	–
151	–1	–
152	–1	–
153	–1	–
154	–1	–
155	–1	–
156	–1	–
157	–1	–
158	–1	–
159	–1	–
160	–1	–
161	161	exclamdown
162	162	cent
163	163	sterling
164	64943	fraction
165	165	yen
166	61346	florin
167	167	section
168	36	currency
169	169	quotesingle

Index	Registration Authority Indentifier	Name
170	170	quotedblleft
171	171	guillemotleft
172	61226	guilsinglleft
173	61227	guilsinglright
174	61472	fi
175	61473	fl
176	–1	–
177	61220	endash
178	61232	dagger
179	61233	daggerdbl
180	183	periodcentered
181	–1	–
182	182	paragraph
183	61286	bullet
184	8973	quotesinglbase
185	61224	quotedblbase
186	186	quotedblright
187	187	guillemotright
188	9284	ellipsis
189	61249	perthousand
190	–1	–
191	191	questiondown
192	–1	–
193	193	grave
194	194	acute
195	195	circumflex
196	196	tilde
197	197	macron
198	198	breve
199	199	dotaccent

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Index	Registration Authority Indentifier	Name
200	200	dieresis
201	–1	–
202	202	ring
203	203	cedilla
204	–1	–
205	205	hungarumlaut
206	206	ogonek
207	207	caron
208	61221	emdash
209	–1	–
210	–1	–
211	–1	–
212	–1	–
213	–1	–
214	–1	–
215	–1	–
216	–1	–
217	–1	–
218	–1	–
219	–1	–
220	–1	–
221	–1	–
222	–1	–
223	–1	–
224	–1	–
225	225	AE
226	–1	–
227	227	ordfeminine
228	–1	–
229	–1	–

Index	Registration Authority Indentifier	Name
230	–1	–
231	–1	–
232	232	Lslash
233	233	Oslash
234	234	OE
235	235	ordmasculine
236	–1	–
237	–1	–
238	–1	–
239	–1	–
240	–1	–
241	241	ae
242	–1	–
243	–1	–
244	–1	–
245	245	dotlessi
246	–1	–
247	–1	–
248	248	lslash
249	249	oslash
250	250	oe
251	251	germandbls
252	–1	–
253	–1	–
254	–1	–
255	–1	–

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
(informative)

Compatibility with the installed base

B.1 Compatibility with installed base of glyph procedure interpreters

There is an installed base of approximately one million glyph procedure interpreters which are largely compatible with section 2. In order for a conforming font program to be backward compatible with this installed base of glyph procedure interpreters, the following guidelines should be observed.

B.1.1 GlyphEncrypt property (GLYPHENCRYPT)

For compatibility with the installed base of glyph procedure interpreters, the GLYPHENCRYPT property should have the default value of TRUE, and the encryption key should have a value of 4330. The encryption only serves as a barrier to casual illegal copying. There is no provision for other encryption key values or other forms of encryption since the purpose of the font standard is font interchange.

B.1.2 lenIV property (LENIV)

To be compatible with version 23.0 of the PostScript™ interpreter[1] (found in the original Apple® LaserWriter®), the value of LENIV should be set to 4. If compatibility with version 23.0 printers is not necessary, it can be set to 0 or 1 to minimize the size of the glyph procedures.

B.1.3 Hint zone values for ghost stems

If the top and bottom extremities of a glyph are to align at positions defined by the values of the font-level properties BLUEVALUES and OTHERBLUES, and it is desired that this work correctly with the installed base of interpreters, there is a restriction on the allowable values for those stem hints. The hint zones for the ghost stems should be created with a vertical height of 20 or 21 glyph coordinate system units; either is acceptable. They should describe a y-coordinate range that is inside the y-coordinate range of the glyph.

B.1.4 Round stem up (ROUNDSTEMUP)

For compatibility with the installed base of Type 1 font program interpreters, creators of font programs specifying Language Group 1 must also include a ROUNDSTEMUP property in the Glyph Properties property-list (see 2.6.3) with a boolean value of FALSE.

The ROUNDSTEMUP property has been superseded by the LANGUAGEGROUP property in the Typographic Color Properties property-list (see 2.6.2). No reference to the name ROUNDSTEMUP should be made in any font program unless it specifies Language Group 1.

The formal structure of the ROUNDSTEMUP property is

```
roundstemup-property  ::=  roundstemup-name, roundstemup-value
roundstemup-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//ROUNDSTEMUP
roundstemup-value  ::=  FALSE
```

1) PostScript™ is an example of a suitable product available commercially. This information is given for the convenience of users of this International Standard and does not constitute an endorsement by ISO/IEC product.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B.1.5 MinFeature (MINFEATURE)

MINFEATURE can be included in the T1 GLYPH property-list (see 2.6.3) for purposes of backward compatibility with the installed base of glyph procedure interpreters.

The formal structure of the MINFEATURE property is

```
minfeature-property  ::=  minfeature-name, minfeature-value-list
minfeature-name  ::=  STRUCTURED-NAME  -- ISO/IEC 9541-3//MINFEATURE
minfeature-value-list  ::=  16, 16
```

B.1.6 Password (PASSWORD)

For compatibility with the installed base, the PASSWORD property shall be present and have a value of 5839.

B.1.7 Accent Component Table

This feature is not backward compatible with the installed base of glyph procedure interpreters. In order to be compatible with this installed base, the Accent Component Table must be absent, thus defaulting to the default Accent Component Table given in annex A. If a font specifies its own Accent Component Table, and that table includes glyphs from the default table, it must use the names specified in the Name column in annex A.

B.1.8 Interchange format

The following two subclauses give the ASN.1 and SGML definitions for the GLYPHPROPS properties MINFEATURE and ROUNDSTEMUP.

B.1.8.1 ASN.1 interchange format

The following two properties are GlyphProps Properties and belong in the T1-Glyph-Property-List (see 2.9.1.1):

```
minfeature      [5]  IMPLICIT MinFeature-List    OPTIONAL,

roundstemup     [6]  IMPLICIT RndStmUp           OPTIONAL,

MinFeature-List             ::= SEQUENCE OF  {
                                IMPLICIT INTEGER (16),
                                IMPLICIT INTEGER (16) }

RndStmUp                    ::= IMPLICIT BOOLEAN
```

B.1.8.2 SGML interchange format

The following SGML definition replaces the ELEMENT declaration for t1gpprp in 2.9.1.2:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
<!ELEMENT  gpprops  - o  (glncrpt?  & leniv?   & accenlst?  & subrs?   &
                          glplists? & minfetur & rndstmup )
                                                    -- GlyphProc Properties -->
<!ELEMENT  glncrpt  - o  (bool)                    -- glyphencrypt Property -->
<!ELEMENT  leniv    - o  (card)                                   -- lenIV -->
<!ELEMENT  accenlst - o  (accentpr*)                    -- AccentEncoding -->
<!ELEMENT  accentpr - o  (accindx, glyphid)      -- AccentComponentIndex/GlyphID -->
<!ELEMENT  accindx  - o  (int)                        -- Accent Component Index -->
<!ELEMENT  subrs    - o  (glyphprc+)                            -- Subroutines -->
<!ELEMENT  glplist  - o  (glprocpr*)                -- Glyph Procedures List -->
<!ELEMENT  glprocpr - o  (glyphid, glyphprc)      --GlyphID/Glyph Procedure Pair -->
<!ELEMENT  glyphid  - o  (glbname)             -- GlyphID, see ISO/IEC9541-1:1991-->
<!ELEMENT  glyphprc - o  (#PCDATA)                        -- Glyph Procedure -->
<!ELEMENT  glbname  o o  (prefix? , strucnm) +                -- Global Name -->
<!ELEMENT  minfetur - o (int,int)                    -- MINFEATURE "16,16" -->
<!ELEMENT  rndstmup - o (bool)                      -- ROUNDSTEMUP "false" -->
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex C
(informative)

Composite glyphs

C.1 Composite glyphs

Composite glyphs can generally be composed in several places in the overall publishing model/cycle. The Type 1 architecture only allows for composite glyphs to be created within the shape description by using a sequence of glyph operators, as opposed to creating them, for example, at the time of document composition.

C.1.1 Using the siag operator to create composite glyphs

The **siag** operator may be used to link the two component glyphs into a composite glyph. The following is an example of how it may be used. Figure C.1 illustrates the resulting accented glyph composed of the "O" (glyph index 79) and the acute accent (glyph index 194).

If the procedure for constructing the "O" (glyph index 79) is as follows:

46 795 xrpe
(*hint operators omitted*)
35 −16 rmoveto
(*path construction operators omitted*)
closepath
endglyph

And the procedure for the acute accent glyph (glyph index 194) is as follows:

123 360 xrpe
(*hint operators omitted*)
145 577 rmoveto
(*path construction operators omitted*)
closepath
endglyph

Then the procedure for constructing the composite glyph may be written as follows (hinting operators omitted for clarity):

46 795 xrpe
99 338 172 79 194 siag
endglyph

Notice that the **xrpe** operator and its operands must be identical to that of the base glyph.

Figure C.1 illustrates the construction of a composite glyph using the siag operator.

--`,,`-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

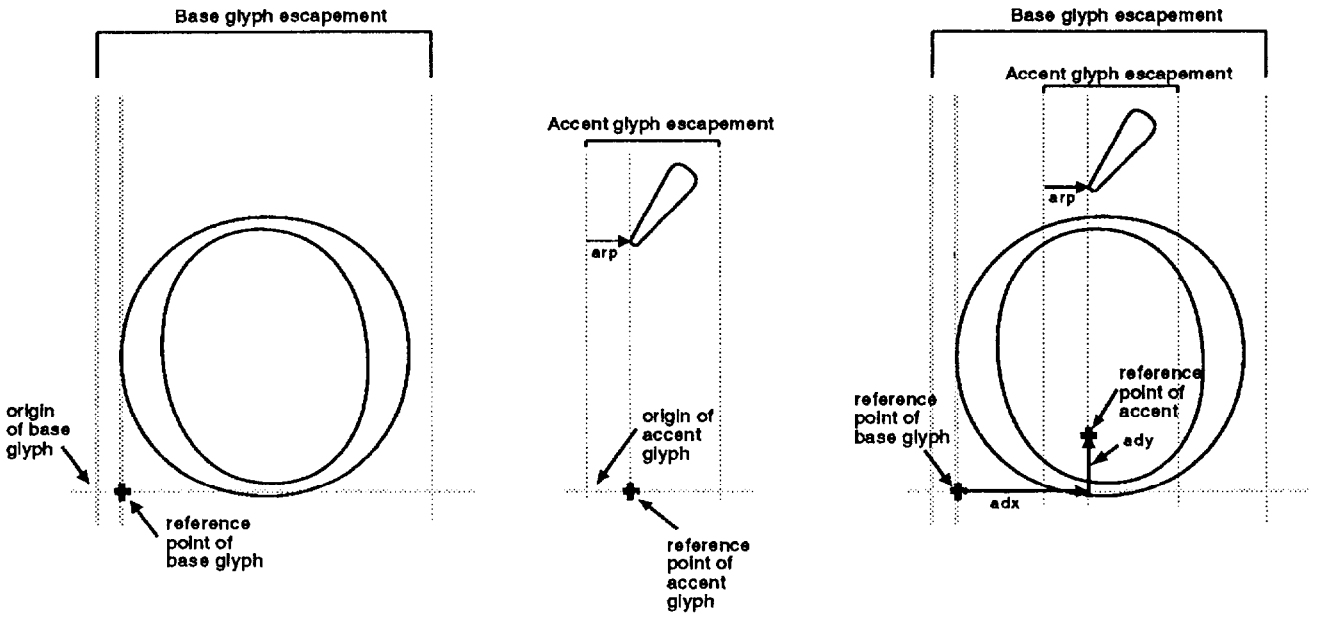


Figure C.1 — Composite glyph constructed with siag operator: component glyphs and final composite glyph

C.1.2 Using subroutines to create composite glyphs

The use of subroutines is the most general and flexible method for constructing composite glyphs. For example, the following procedure would construct a composite glyph for the katakana glyph *Ga* (optional hint operators are not shown):

```
30 800 xrpe
170 620 rmoveto
7 callsubr
540 80 rmoveto
8 callsubr
endglyph
```

Figure C.2 illustrates the subpaths constructed by each subroutine and the resulting glyph.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

© ISO/IEC

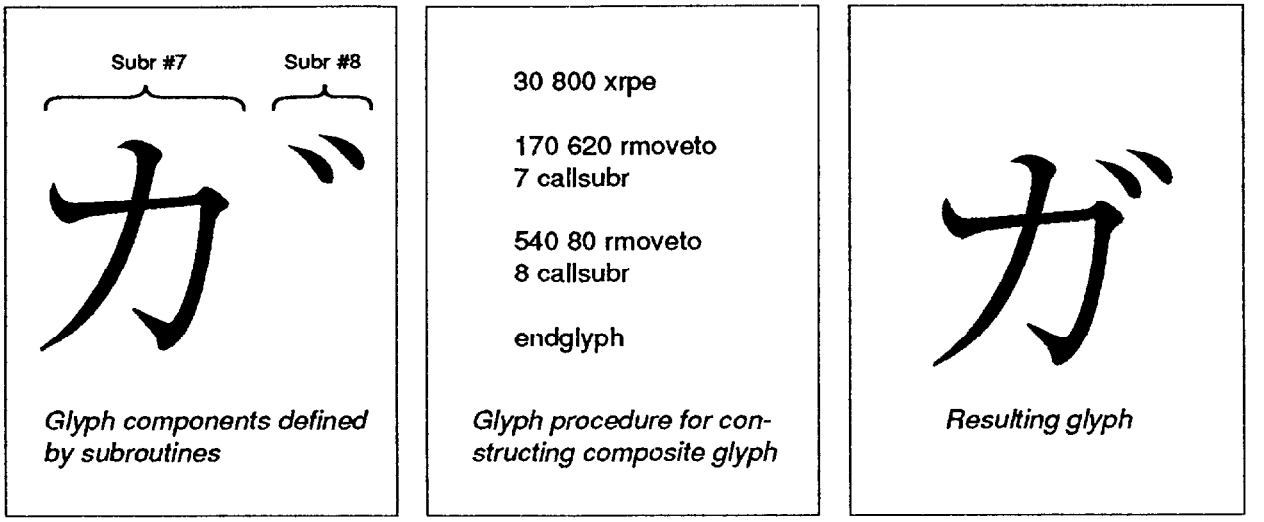


Figure C.2 — Construction of a composite glyph using subroutines

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex D
(informative)

Interchange sample

To illustrate the representation of operator values in a glyph procedure, consider the following example of a block letter "C" (illustrated in figure D.1). In the glyph coordinate system, this letter measures 700 units by 700 units. Its escapement is 800 units (horizontal), and it is centered within this escapement; thus its reference point has a value of 50 units. Each stem is 100 units wide. This example shows how to declare these glyph-level hint zones for these stems.

The glyph procedure definition begins with a representation in glyph coordinate system integer coordinates:

```
50 800 xrpe 0 100 vstem 0 100 hstem 600 100 hstem
0 0 rmoveto 700 0 rlineto 0 100 rlineto −600 0 rlineto
0 500 rlineto 600 0 rlineto 0 100 rlineto −700 0 rlineto
closepath
```

NOTE 47 The values of the **vstem** and **hstem** operands are relative to the reference point.

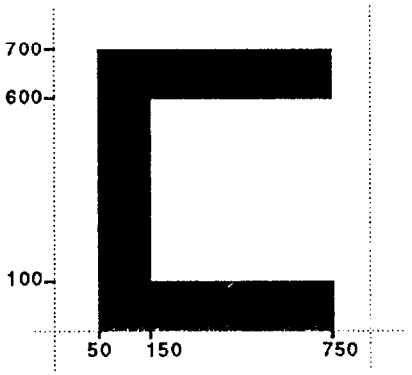


Figure D.1 — Simplified glyph for encoding example

The initial **rmoveto** operator (or its equivalent) is required, as the **xrpe** operator only sets the *currentpoint* but does not actually place that point in the glyph path. In this example, the first point on the path is the same as the reference point, thus the two zero operands to the **rmoveto** operator. In other glyph procedures, the initial **rmoveto** point need not be the same as the reference point.

Note that there are many horizontal and vertical **rlineto** commands. Modify them to **hlineto** and **vlineto** operators for space efficiency. Finish the glyph with an **endglyph** operator.

```
50 800 xrpe 0 100 vstem 0 100 hstem 600 100 hstem
0 hmoveto 700 hlineto 100 vlineto −600 hlineto
500 vlineto 600 hlineto 100 vlineto −700 hlineto
closepath endglyph
```

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Express the integers according to glyph procedure number representation:

189 249 180 xrpe 139 239 vstem 139 239 hstem 248 236 239 hstem
139 hmoveto 249 80 hlineto 239 vlineto 252 236 hlineto
248 136 vlineto 248 236 hlineto 239 vlineto 253 80 hlineto
closepath endglyph

Convert the operators according to glyph procedure operator representation:

189 249 180 13 139 239 3 139 239 1 248 236 239 1
139 22 249 80 6 239 7 252 236 6
248 136 7 248 236 6 239 7 253 80 6
9 14

For purposes of illustrating this example, this sequence of numbers is rewritten in octet notation:

11/13 15/09 11/04 00/13 08/11 14/15 00/03 08/11 14/15 00/01 15/08
14/12 14/15 00/01 08/11 01/06 15/09 05/00 00/06 14/15 00/07 15/12
14/12 00/06 15/08 08/08 00/07 15/08 14/12 00/06 14/15 00/07 15/13
05/00 00/06 00/09 00/14

If the GLYPHENCRYPT property value is TRUE, this octet string must be encrypted. In glyph procedure encryption, the initial key for the variable R is 4330 (decimal). The number of random octets, n, is set within the font. By default, n is 4. However, if a property name lenIV is present in the General Properties list, then n is the value associated with lenIV.

Glyph procedure encryption imposes no restrictions on the values of the initial ciphertext octets.

To encrypt the above example, the following procedure should be used. The 37 octets shown above constitute the plaintext of the glyph procedure. Generate four random plaintext octets to insert at the front of this plaintext glyph procedure. This example uses four zeros (for ease of explanation), resulting in this plaintext:

00/00 00/00 00/00 00/00 11/13 15/09 11/04 00/13 08/11 14/15 00/03

08/11 14/15 00/01 15/08 14/12 14/15 00/01 08/11 01/06 15/09 05/00

00/05 14/15 00/07 15/12 14/12 00/06 15/08 08/08 00/07 15/08 14/12

00/06 14/15 00/07 15/13 05/00 00/06 00/09 00/14

Apply glyph procedure encryption to produce the following 41 octets of ciphertext expressed as an octet string:

01/00 11/15 03/01 07/00 04/15 10/11 05/11 01/15 00/03 15/09 11/06

08/11 01/15 03/09 10/06 06/05 02/01 11/01 08/04 01/15 01/04 08/01

06/09 07/15 08/14 01/02 11/07 15/07 13/13 13/06 14/03 13/07 02/04

08/13 09/06 05/11 01/12 13/04 05/14 02/01 01/04

Finally, express this encrypted glyph procedure in binary form.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Alphabetical index

A

Accent Component Table 2.7.3.1.4, B.1.7
ACCENTENCODING 2.6.3.3
Alignment position 2.4.4
Alignment zone 2.4.8
Arithmetic operator 2.7.3.4, 2.7.3.4.1
— div 2.7.3.4.1
ASN.1 2.9.1.1

B

Bézier curve 2.4.15
bitmap 1.5
BLUEFUZZ 2.6.2.7
BLUESCALE 2.6.2.5
BLUESHIFT 2.6.2.6
BLUEVALUES 2.6.2.1

C

callsubr 2.7.3.5.1
callutilsubr 2.7.3.5.3
ciphertext 2.2.1
closepath 2.7.3.2.1
Composite glyphs 2.8.5, 2.8.5.1, 2.8.5.2, C.1, C.1.2
— siag operator 2.8.5.1
— Subroutines 2.8.5.2
Current path 2.7.1.4.3
current point 2.2.2, 2.7.1.4.1

D

Data types 2.7.2
div 2.7.3.4.1
Dot section 2.7.1.4.6
dotsection 2.7.3.3.1

E

encoding 1.1
encryption key 2.2.3
endglyph 2.7.3.1.3
Escapement 2.7.1.4.2

F

FAMILYBLUES 2.6.2.3
FAMILYOTHERBLUES 2.6.2.4
Flat position 2.4.5
Flex mechanism 2.4.13, 2.8.3
font program 2.2.4
font-level alignment zone 2.4.9
FORCEBOLD 2.6.2.8.5

G

ghost stem 2.2.5
ghost stems 2.4.9, B.1.3
Glyph coordinate system 2.4.1
Glyph operator interchange representation 2.9.2.2
glyph procedure 2.2.6
Glyph procedure encryption 2.9.2.3
glyph procedure interpreter 2.2.7, 2.4.3
Glyph procedure interpreter model 2.5
Glyph procedure language 2.4.2
Glyph procedure operators 2.7.3
Glyph procedure semantics 2.7
Glyph procedures 2.7.1.1
GLYPHENCRYPT 2.6.3.1, B.1.1
GLYPHPROC 2.6.3.5.2
GLYPHPROCPROPS 2.6.3.5
GLYPHPROPS 2.6.3
GNAME 2.6.3.5.1

H

hint 2.2.8
Hint operators 2.7.3.3, 2.7.3.3.1, 2.7.3.3.2, 2.7.3.3.3, 2.7.3.3.4, 2.7.3.3.5
— dotsection 2.7.3.3.1
— hstem 2.7.3.3.2
— hstem3 2.7.3.3.3
— vstem 2.7.3.3.4
— vstem3 2.7.3.3.5
Hint substitution 2.4.14, 2.8.2
Hint zone 2.4.10
hints 1.5, 2.3, 2.4.9
hlineto 2.7.3.2.2
hmoveto 2.7.3.2.3
Horizontal stems 2.7.1.4.4
hstem 2.7.3.3.2
hstem3 2.7.3.3.3
hvcurveto 2.7.3.2.4

I

Implicit results stack 2.7.1.5
Interchange format 2.9, 2.9.1.1, B.1.8
— ASN.1 2.9.1.1
Interchange format for glyph procedures 2.9.2
Interchange sample D
Interpreter 2.7.1.2

L

LANGUAGE GROUP 2.6.2.8.6
LENIV 2.6.3.2, B.1.2

M

MINFEATURE B.1.5

O

Object types 2.7.2.1, 2.7.2.1.1, 2.7.2.1.2
— Integer 2.7.2.1.1
— Number 2.7.2.1.2
Operand list 2.7.1.3
OTHERBLUES 2.6.2.2
outline 1.5
overshoot 2.2.9
Overshoot position 2.4.6
Overshoot suppression 2.4.7

P

PAINTTYPE 2.6.1.2
PASSWORD 2.6.1.1, B.1.6
path 2.2.10
Path construction operators 2.7.3.2, 2.7.3.2.1, 2.7.3.2.2, 2.7.3.2.3, 2.7.3.2.4, 2.7.3.2.5, 2.7.3.2.6, 2.7.3.2.7, 2.7.3.2.8, 2.7.3.2.9, 2.7.3.2.10, 2.7.3.2.11
— closepath 2.7.3.2.1
— hlineto 2.7.3.2.2
— hmoveto 2.7.3.2.3
— hvcurveto 2.7.3.2.4
— rlineto 2.7.3.2.5
— rmoveto 2.7.3.2.6
— rrcurveto 2.7.3.2.7
— setcurrentpoint 2.7.3.2.11
— vhcurveto 2.7.3.2.8
— vlineto 2.7.3.2.9
— vmoveto 2.7.3.2.10
Path direction 2.4.11
plaintext 2.2.11
properties 2.3
— general 2.3
— glyph procedure 2.3
— typographic color 2.3

R

rasterization 2.2.12
Reference point 2.4.12
region 2.2.13
return 2.7.3.5.2
retval 2.7.3.5.4
rlineto 2.7.3.2.5
rmoveto 2.7.3.2.6
ROUNDSTEMUP B.1.4
rpe 2.7.3.1.1
rrcurveto 2.7.3.2.7

S

segment 2.2.14
setcurrentpoint 2.7.3.2.11
siag 2.7.3.1.4, 2.8.5.1, C.1.1
snap 2.2.15
Starting and finishing operators 2.7.3.1, 2.7.3.1.1, 2.7.3.1.2, 2.7.3.1.3, 2.7.3.1.4
— endglyph 2.7.3.1.3
— rpe 2.7.3.1.1

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— siag 2.7.3.1.4
— xrpe 2.7.3.1.2
State variables 2.7.1.4, 2.7.1.4.1, 2.7.1.4.2, 2.7.1.4.3, 2.7.1.4.4, 2.7.1.4.5, 2.7.1.4.6
— CurrentPath 2.7.1.4.3
— CurrentPoint 2.7.1.4.1
— DotSection 2.7.1.4.6
— Escapement 2.7.1.4.2
— HorizontalStems 2.7.1.4.4
— VerticalStems 2.7.1.4.5
STDHW 2.6.2.8.1
STDVW 2.6.2.8.2
STEMSNAPH 2.6.2.8.3
STEMSNAPV 2.6.2.8.4
STEMWIDTHPROPS 2.6.2.8
subpath 2.2.16
Subroutine operators 2.7.3.5, 2.7.3.5.1, 2.7.3.5.2, 2.7.3.5.3, 2.7.3.5.4
— callsubr 2.7.3.5.1
— callutilsubr 2.7.3.5.3
— return 2.7.3.5.2
— retval 2.7.3.5.4
Subroutines 2.8, 2.8.5.2, C.1.2
SUBRS 2.6.3.4, 2.8.4

T

Type 1 glyph shape concepts 2.4, 2.4.1, 2.4.2, 2.4.3, 2.4.4, 2.4.5, 2.4.6, 2.4.7, 2.4.8, 2.4.9, 2.4.10, 2.4.11, 2.4.12, 2.4.13, 2.4.14, 2.4.15
— Alignment position 2.4.4
— Alignment zone 2.4.8
— Bézier curve 2.4.15
— Flat position 2.4.5
— Flex mechanism 2.4.13
— Glyph coordinate system 2.4.1
— Glyph procedure interpreter 2.4.3
— Glyph procedure language 2.4.2
— Hint substitution 2.4.14
— Hint zone 2.4.10
— Hints 2.4.9
— font-level 2.4.9
— glyph-level 2.4.9
— Overshoot position 2.4.6
— Overshoot suppression 2.4.7
— Path direction 2.4.11
— Reference point 2.4.12
typographic color 2.2.17
T1COLORPROPS 2.6.2
T1GENERAL 2.6.1
T1SHAPES 2.6

U

UNIQUEID 2.6.1.3
Utility subroutine 0 2.8.1.1
Utility subroutine 1 2.8.1.2
Utility subroutine 2 2.8.1.3
Utility subroutine 3 2.8.1.4
Utility subroutines 2.8.1

V

Vertical stems 2.7.1.4.5
vhcurveto 2.7.3.2.8
Virtual machine 2.7.1
vlineto 2.7.3.2.9
vmoveto 2.7.3.2.10
vstem 2.7.3.3.4
vstem3 2.7.3.3.5

X

xrpe 2.7.3.1.2

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

© ISO/IEC

ICS 35.240.30

Descriptors: data processing, text processing, information interchange, character sets, alphanumeric character sets, character subsets, representation of characters, shape.

Price based on 60 pages

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale