SBL Hebrew Font User Manual

Font version 1.12
Manual version 1.04, November 2005

Prepared by John Hudson, Tiro Typeworks,
for the Society of Biblical Literature

Manual version 1.04
November 2005

It is recommended that all users of the SBL Hebrew font read this manual carefully, even if they have used earlier versions of the font and are familiar with encoding requirements and layout behaviour discussed in earlier versions of the manual. There are a number of important changes in the recommended encoding of some characters, some of which may require updates to existing documents made with earlier versions of the font. Since the first public version of the font shipped in 2003, there have been a number of clarifications and additions to the Unicode codepoint range for Hebrew. The latest version of the font, v1.12, supports all Hebrew characters in the recently released version 4.1 of the Unicode Standard. This version of Unicode provides a couple of important additions for Biblical Hebrew (notably, a codepoint for *nun hafukah,* allowing deprecation of the hack used to access this as a glyph variant in previous versions of the SBL Hebrew font). Discussions with members of the Unicode Technical Committee and other interested parties have also clarified the recommended encoding for the upper and lower *puncta extraordinaria* and other marks. Sections of this manual that relate to such additions or other changes from the behaviour of previous font versions are marked with a red line in the right margin, thus.

Getting started

If you have not already downloaded the latest version of the SBL Hebrew font, you should do so now. See page 22 for download information and links to other useful websites.

Please take some time to review this manual. SBL Hebrew is a complex font that makes use of new technologies that may be unfamiliar to you if you have not worked with Unicode encoded text or font files before. Many of the lessons you may have learned from using other Biblical Hebrew fonts and software may not apply, and it may be necessary to develop some new work habits.

Installation

The SBL Hebrew font is designed for use primarily with Windows 2000, Windows XP and later Microsoft operating systems. It may also be used cross platform in applications that natively support appropriate Unicode text processing and OpenType Layout. To install the font on a Windows system, open the Fonts folder and drag and drop the *SBL_Hbrw.ttf* file onto the folder. The Fonts folder can be opened either by navigating in Windows Explorer to

```
c:\Windows\Fonts
```

(where *c:* is the root folder containing your Windows system files), or by opening the Fonts folder from the Control Panel, accessible from the Start button Settings menu.

The SBL Hebrew font can also be installed on older versions of Windows, but because the font uses a pure Unicode encoding, not an 8-bit codepage, it may not work consistently in all applications.

It is also possible to install the font on Linux and other open source systems using the FreeType library, and on Mac OS X. Please see pages 17–18 for information about support on non-Windows systems.

File name and icon

Although SBL Hebrew is an OpenType format font, it has a .ttf file name extension and will display the regular TrueType icon, rather than the distinctive OpenType icon. The .ttf extension allows the font to be recognised on older versions of Windows (although these only support unvocalised Hebrew text). The regular TrueType icon is displayed only because the font does not contain a digital signature. Note that the different icons available for TrueType OT fonts do not indicate anything about the presence or absence of OpenType Layout tables for glyph substitution or positioning, both of which are present present in the SBL Hebrew font. See pages 4–5 for more information.

Introduction

Most fonts do not come with user manuals, and most do not need them. SBL Hebrew is a complex font that uses new encoding and layout technologies, and this manual explains these technologies to help you get the most out of your new font. This manual also discusses known issues relating to these technologies, particularly current levels of operating system and application support, and the complicated but important issue of text normalisation. Although you can install the SBL Hebrew font on most current operating systems, and can immediately begin working with it, please take time to review this document. Understanding issues like optimal character ordering will help ensure that the SBL Hebrew correctly displays your text.

The font format

This section is probably the least essential reading in this manual. This section explains the background to the development of the technology used in the SBL Hebrew font, and explains exactly what kind of font this is in technical terms.

SBL Hebrew is a TrueType-flavour OpenType font.

The original TrueType font format was developed by Apple Computers and released in the early 1990s. It was quickly licensed by Microsoft Corp. and is the system font format for Apple and Microsoft operating systems (the packaging of the format differed on the two systems prior to the introduction of Apple's Mac OS X, which can install Windows TT fonts). A TrueType font is a collection of tables, each containing information specific to a particular aspect of the font. For example, the *glyf* table contains outline information, the *cmap* table maps glyphs in the font to character codes for text entry and storage, and so forth.

The OpenType font format is an extension of the TrueType format, jointly developed by Microsoft Corp. and Adobe Systems. An OpenType font has the same table structure as a generic TrueType font, but with the option to include additional tables. There are two key components to the OpenType extensions: PostScript outline data and OT Layout data. The first of these determines the 'flavour' of a font—that is, the kind of outline and rendering technology used to 'paint' text. As noted above, SBL Hebrew is TrueType-flavour, meaning that it uses the original TrueType outline format and rendering technology rather than PostScript. The second key component, OpenType Layout data, is essential to the correct rendering of complex scripts like Hebrew, and much of this manual is concerned with this technology and how it renders Biblical Hebrew. The SBL Hebrew font contains two kinds of OpenType Layout data: glyph substitution and glyph positioning. These are stored in the OpenType optional tables *GSUB* and *GPOS*.

For these and other online documents, please see the list of URLs in Appendix C, p.22

For more information about the OpenType font format, see the specification and other material on the Microsoft or Adobe websites. The Microsoft introductory essay *Windows Glyph Processing,* which also discusses other aspects of complex script shaping referred to in

this manual, might be a good place to start. None of this material is essential to being able to use the SBL Hebrew font, but it will help you better understand that technologies on which the font relies.

Unicode text encoding

This section explains, in simplified terms, what the Unicode Standard is, and the implications of this encoding standard for Biblical Hebrew and for users of the SBL Hebrew font.

The SBL Hebrew font uses standard Unicode character codes to encode Hebrew letters and marks. Unicode is an international character encoding standard that provides a single unique code for every semantic character necessary to encode plain text in supported scripts and languages. Today, most of the world's major scripts and languages are supported by Unicode, and recent efforts have led to the encoding of numerous historical scripts. Because every character in Unicode has a unique code, rather than sharing codes across multiple codepages or being assigned to different codes *e.g.* on Windows and Mac operating systems, text encoded in Unicode can be safely exchanged between different operating systems and applications that employ Unicode. The benefit of such a standard to scholarship, where authors and publishers are often using different software, should be obvious.

An 8-bit encoding is one that uses a single byte of computer memory or storage for each character in text. 8-bit encodings are limited to 256 characters, which explains the need for multiple encodings.

In the past, Hebrew text was supported on different platforms, and even between different applications, using a variety of standard and not-so-standard 8-bit encodings. Because the fonts that supported these encodings tended to be 'dumb' fonts, *i.e.* without built-in layout intelligence, often multiple fonts would be needed to correctly display complex texts such as found in Biblical scholarship. This, of course, increased the likelihood that text produced with these encodings could not be reliably and accurately exchanged between systems and applications, because correct display relied on not only knowing which encoding standard was used but also which font was used where. Unicode and OpenType solve these problems by using a unique code for each Hebrew consonant and mark, and employing layout intelligence to map from the encoded characters to the appropriate arrangement of glyphs to display a given text. [See the examples on the following pages.]

While the benefit of Unicode is easy to see, it does require that the SBL Hebrew font be used in a Unicode text encoding environment, *i.e.* in systems and applications that use Unicode character codes when storing, manipulating and displaying text. The font does not contain an 8-bit codepage—only Unicode—so no guarantees can be made about its performance in non-Unicode environments, [Note, however, that some applications will internally map from Unicode characters in a font to 8-bit codes used internally by the application. In this case some text may display correctly with the SBL Hebrew font, although the text remains subject to the typical exchange problems of 8-bit text.] It may be that your current document publish-

ing workflow involves non-Unicode applications, and this will limit, for the time being, the usefulness of the SBL Hebrew font in your work. During the past few years, Unicode has become the dominant encoding standard for most major operating systems, and is, for example, specified by the World Wide Web consortium as the default encoding of XML documents. It is very likely that the makers of your present software will be updating their applications to handle Unicode, and they may be able to give you estimates on when suitable upgrades will be available. If you are tied to software that is not being updated to handle Unicode soon, or if you simply cannot wait and like the *design* of the SBL Hebrew typeface, it may be possible to arrange to have custom fonts made for specific 8-bit encodings, or to make your own. For information on the kind of modifications that are permitted, or for contact information, please see the font license in Appendix A, page 19.

Uniscribe

This section explains the role of the Microsoft Unicode Script Processor (Uniscribe) in shaping complex scripts. The most important thing to note in this regard is that different versions of Uniscribe will produce different results for some Hebrew mark sequences.

The layout intelligence in OpenType fonts relies on system or application support for basic linguistic shaping. In simple terms, systems and applications deal with characters, and fonts deal with glyphs, *i.e.* with the visual representation of characters. Complex scripts like Hebrew require systems and applications to be aware of right-to-left and bidirectional layout and of character properties that distinguish *e.g.* consonants from combining marks. Some applications will have built in support for such things, while others will rely on standard system components. Microsoft's Unicode Script Processor—commonly referred to as Uniscribe—is a standard system component that includes shaping engines for various scripts, including Hebrew. Uniscribe works directly with OpenType fonts, such as SBL Hebrew, built according to Microsoft specifications. An application that uses Uniscribe will make calls to it as text is entered and edited; Uniscribe processes the input characters, and applies the layout intelligence in the font accordingly.

Note that applications using text processing calls on systems older than Windows 2000 may not be Uniscribe clients.

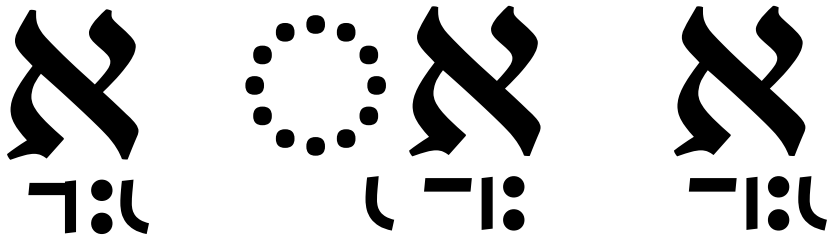
If you are using an application that utilises Uniscribe—these include the Microsoft Office Suite on Windows, Internet Explorer, and any other app that makes standard system calls for text input and output—you will find working with Biblical Hebrew text very easy and the SBL Hebrew font will automatically correctly display most text. I say most text, because there is always the possibility to create sequences of consonants plus marks that are linguistically non-standard and which cannot be correctly resolved by the layout features in the font. The SBL Hebrew font has been tested with every combination of consonant plus mark(s) that occurs in the Michigan-Claremont Old Testament text. The results of this testing are available as a 121-page Acrobat document from the Society of Biblical

This document is also available from http:www.tiro.com/resources/hebrew/

Literature Hebrew font website. The document shows every unique consonant plus mark(s) sequence that occurs in the text along with following consonant context.

Software developers producing software for use in Biblical scholarship can arrange for a redistribution license for Uniscribe. This enables them to bundle more recent versions of Uniscribe for utilisation by their software. Contact Microsoft Typography for more information.

Different versions of Uniscribe ship with and are used by different applications and system versions, which means that rendering results may vary. We have been fortunate, during the development of SBL Hebrew, to test the fonts with unreleased versions of Uniscribe that will ship with upcoming software. The new versions of Uniscribe have been used to confirm that all consonant plus mark(s) sequences in the Michigan-Claremont text will be correctly rendered *when these versions of Uniscribe are employed.* Because SBL Hebrew has, in this respect, been designed for next generation technology, some sequences will display incorrectly in systems and applications using older versions of Uniscribe. It is important to note that these sequences may be correctly encoded, and that documents with display problems are not erroneous. Typically, when a sequence of marks cannot be correctly rendered by Uniscribe, the shaping engine will insert a dotted-circle, and the mark or marks that cannot be applied to the preceding consonant will be applied to the circle. The example below shows three possible levels of display for a unique sequence of consonant plus marks from Job 7:11 (*alef* + *hataf patah* + *meteg* + *dehi*).



C. *Incorrect* B. *Incorrect* A. *Correct*

A. This example shows the correct rendering applied by Uniscribe version 1.468.4015.0, a recent beta version of Uniscribe. This and all later versions of Uniscribe (including 1.471.4063.0, which ships with MS Office 2003) can be expected to correctly render this and other consonant plus mark sequences.
B. This example shows incorrect rendering applied by Uniscribe version 1.325.2180.1; this is the version of Uniscribe that ships with Windows 2000. As you can see, this version of Uniscribe is unable to correctly apply *dehi* to a consonant that already carries a combination of *hataf* vowel and *meteg,* so inserts a dotted circle.
C. This very incorrect example shows the same sequence crudely displayed by an application that does not use Uniscribe and cannot implement the glyph positioning intelligence in the SBL Hebrew

font. The marks are blindly centered below the consonant, but they collide and do not interract correctly. In some applications, the marks may not even be centered below the consonant, but will cluster between it and the next letter.

Note that in the two examples rendered by Uniscribe the *meteg* combines with the *hataf patah,* forming a mark ligature in which the *meteg* appears between the two parts of the vowel. For more information about this and other aspects of *meteg* handling, see the section of this manual on page 13.

The normalisation issue

This section explains an important problem in the encoding of Hebrew in the Unicode Standard. This problem will not affect all text, but can be a serious problem that users need to be aware of.

Although implementation of the Unicode Standard is generally a boon to scholars working with texts in complex scripts, there is an unfortunate and quite serious problem in the current encoding of Hebrew. This involves the canonical combining class assignments that are used when text is normalised. Normalisation is a process by which sequences of characters in text that can be variously encoded but are *semantically identical* are treated as identically encoded. This can frequently involve the reordering of a sequence of characters. Consider, as an example, this combination of consonant plus marks that occurs in 1 Ch 13:13. This combination could be encoded in six different ways, and each would result in exactly the same visual representation and the same semantic meaning for the reader:

1. *tet* + *dagesh* + *tsere* + *zaqef gadol*
2. *tet* + *dagesh* + *zaqef gadol* + *tsere*
3. *tet* + *tsere* + *dagesh* + *zaqef gadol*
4. *tet* + *tsere* + *zaqef gadol* + *dagesh*
5. *tet* + *zaqef gadol* + *dagesh* + *tsere*
6. *tet* + *zaqef gadol* + *tsere* + *dagesh*

The SBL Hebrew font is able to correctly render any of these sequences (although most users familiar with Hebrew would agree that the *dagesh* should, logically and linguistically, precede the vowel and the cantillation mark, and most would also agree that the vowel should precede the cantillation mark). If you consider that any combination of consonant plus three marks can be encoded in six different ways, it is easy to realise how even a fairly short word of five or six consonants with all their marks could be encoded in many dozens of different ways. Normalisation is important because it provides a mechanism for all these possible permutations of mark ordering to be resolved to a single canonical order. This is most important when a text not only needs to be displayed but also needs to searched, sorted or spellchecked. If a search algorithm had to look for fifty or more possible and equivalent spellings of a single word, it would

be extremely inefficient and slow. So normalisation is applied to reorder every equivalent sequence of characters into a single and consistent order.

Normalisation is achieved by giving every mark a canonical combining class. This is a number that indicates how close to the base character (the consonant, in the case of Hebrew) each mark should be ordered and which marks may be reordered relative to each other. Some canonical combining classes contain only individual characters, indicating that these can always be reordered relative to characters with different classes; some classes contain multiple characters, indicating that these *cannot* be reordered relative to each other. If two or more characters are included in the same class, this means that their relative ordering is semantically meaningful and not equivalent; therefore, they must not be reordered or the meaning will be lost.

So far, so good, but when the Hebrew script was encoded in the Unicode Standard, combining classes were assigned in a way that failed to take into account the many peculiarities of Biblical texts. This has resulted in every Hebrew vowel being assigned its own canonical combining class, meaning that combinations of two vowels applied to a single consonant may be reordered during normalisation. This is not a problem for modern Hebrew spelling, but can be disastrous for Biblical texts due to textual conventions such as the tendency to record changing pronunciation by applying new vocalisation but preserving the original consonant structure of words. A good—and important—example is the change in pronunciation of *y^{e}rušālēm* to *y^{e}rušālayim,* which required the Masoretes to add *hiriq* between the *lamed* and the final *mem* in order to approximate the new pronunciation as *y^{e}rušālaim,* while preserving the consonant structure of the ancient manuscripts. The final four characters of this word are correctly encoded in the order *lamed* + *patah* + *hiriq* + *final mem,* and the word is correctly displayed יְרוּשָׁלַֽם; however, the faulty canonical combining classes for *patah* and *hiriq* in Unicode cause *hiriq* to always precede *patah* when reordering due to normalisation is applied. The resulting sequence *lamed* + *hiriq* + *patah* + *final mem* is textually incorrect, and it cannot be correctly displayed by the SBL Hebrew font: יְרוּשָׁלַם (note collision of vowels under *lamed*). Unicode normalisation can easily break Biblical Hebrew text.

The good news is that most software does not automatically apply normalisation, and software developers familiar with Biblical Hebrew are likely to be aware of the problem. There remains a risk, however, especially when documents are being exchanged between different platforms or published on the Internet, that a piece of software beyond the original author or editor's control—a web browser

Note that the Unicode Standard includes multiple normalisation forms, i.e. *different ways of normalising text, and not all of these involve reordering of marks. The form that is of concern for Biblical Hebrew encoding is Normalisation Form C, which does reorder marks.*

For more thorough discusion of this example, see Tov, Emanual. Textual criticism of the Hebrew Bible. *2nd edition, Minneapolis, 1992. p43.*

Here are the Hebrew words from this example enlarged. Correct, before normalisation:

יְרוּשָׁלַֽם

Inorrect, after normalisation:

יְרוּשָׁלַם

on the receiving end of an electronic document, for example—may apply normalisation. This is, of course, not a font issue *per se;* it is a text encoding issue that can affect any document and result in textual error, incorrect display, or both. It is very likely that future rendering engines for Hebrew may perform corrective mark ordering during rendering, to ensure that normalised sequences can be correctly rendered by fonts built to a common specification. This will remove many of the possible display problems associated with Unicode normalisation. However, there will remain a number of cases in which the results of normalisation may display 'correctly' but in which the meaning of the text is changed. For instance, if *meteg* is positioned at the right side of a vowel, it may be moved to the left side during normalisation: both positions will render correctly, but only one is textually correct. So mechanisms are necessary to guard against some mark reordering during normalisation.

Software developers who need to apply normalisation to text to facilitate efficient searching can, of course, avoid the problems of the Unicode mark reordering by using a custom normalisation routine that does not reorder Hebrew vowels. Combining classes for such a custom routine, based on the mark ordering recommendations of the next section, are suggested in Appendix B, page 20. However, this does not address the issue of Unicode normalisation being applied 'downstream' of document creation, *e.g.* in web browsers.

If you are using the Tiro Biblical Hebrew keyboard v1.2, you can insert the CGJ *character manually using the key combination* [ctrl+alt+7]. *If you are using the SIL Biblical Hebrew keyboard, you can insert the* CGJ *character using the key combination* [shift+ctrl+alt+P]

The Unicode Technical Committee recommends the use of the Combining Grapheme Joiner (CGJ, U+034F) character as a control character to override mark reordering during normalisation, and the SBL Hebrew font has been updated to handle this mechanism. Using this mechanism, the problem case of mark reordering discussed on page 9 would be resolved by inserting the CGJ character between the patah and hiriq marks: lamed + patah + CGJ + hiriq + final mem. The CGJ character can be inserted between any two adjacent mark characters that may be subject to reordering during normalisation, and will prevent this reordering from happening. Software developers are currently discussing ways in which this can be implemented automatically, so that the burden is not on document authors to identify situations in which mark reordering might occur and to manually insert CGJ at these places. It may be some time before such solutions become widely implemented.

See the last paragraph on page 13 for additional examples of mark reordering and CGJ *insertion.*

If you are not concerned about mark reordering, e.g. because you are working in a completely closed text environment that does not employ normalisation or which uses a custom normalisation rather than the Unicode one, you do not need to worry about the CGJ character in this regard. If you are producing documents that may be published to the Internet or in other circumstances in which you do

not have control of normalisation, CGJ provides a means for you to prevent unwanted reordering. If you are working with specific Bible passages, and are concerned about normalisation reordering of marks, you may find it helpful to check the chapter and verse to see what words may be affected in a way that changes the recommended mark order discussed in the following sections of this manual. To assist in this, we will be producing a document for each Bible book, identifying such words. These documents will be found online at *http:www.tiro.com/resources/hebrew*

Many thanks to Eli Evans at Libronix for help in preparing these documents.

Note that not all mark reordering results in broken rendering or textual ambiguity. As noted above, some rendering engines may correct reordering during display, further reducing the number of instances in which CGJ needs to be inserted; the most important use of CGJ is in instances where the reordered text may render 'cleanly' but where textually important distinctions are lost, e.g. where a *meteg* moves from one side of a vowel mark to another.

Mark ordering

This section discusses recommended ordering for sequences of consonants with multiple marks. This is one of the more important sections of this manual.

As discussed in the previous section, the SBL Hebrew font can correctly display equivalent sequences of marks applied to consonants in different orders. The font cannot correctly display *every* possible permutation, especially when more than two marks are involved, but it seldom matters, for instance, whether a below vowel or above cantillation mark is applied to a consonant first. That said, there are many benefits to establishing consistent habits in ordering marks when creating documents. By following the recommended mark ordering outlined in the table on the next page, you will avoid entering sequences that cannot be correctly displayed—unless, of course, they are beyond the general ability of the SBL Hebrew font to render the Old Testament text—, and you will also make searching for Hebrew words and phrases easier for yourself and colleagues with whom you exchange electronic documents.

The basic principles of the mark ordering recommendation is that marks affecting the pronunciation of consonants are applied first; then the *holam* mark; then below marks, vowels and consonants, as they occur from right-to-left *except* the prepositive marks *yetiv* and *dehi,* which are applied after other low marks because they are positioned relative to them; then above marks, including metatextual marks such as the masora circle, as they occur from right-to-left *except* the postpositive marks *pashta, telisha qetana* and *zinor.* The table of marks on the next page shows the order in which they should be entered (note that this corresponds, also, to the recommendations for custom normalisation in Appendix B, page 20).

Table 1
Recommended mark ordering

1	Base consonant	שׁ
2	*shin dot*	שׁ
	sin dot	שׂ
3	*dagesh/mapiq*	שּׁ
4	*rafe*	שׁ֫
5	*holam*	שׁ
6	Below marks from the following group as the occur from right-to-left:	
	sheva	שְׁ
	hataf segol	שֱׁ
	hataf patah	שֲׁ
	hataf qamats	שֳׁ
	hiriq	שִׁ
	tsere	שֵׁ
	segol	שֶׁ
	patah	שַׁ
	qamats (gadol)	שָׁ
	qamats qatan	שָׁ
	qubuts	שֻׁ
	meteg[1]	שֽׁ
	atnah	שׁֈ
	atnah hafukh	שׁ֮
	tipeha	שֺׁ
	tevir	שׁ֛
	munah	שׁ֣
	mahapakh	שׁ֤
	merkha	שֽׁ
	merkha kefula	שׁ֒
	darga	שׁ֓
	yerah ben yomo	שׁ֪
	low punctum extra.[2]	שׁ֙
7	Prepositive below marks from the following group:	
	yetiv	שׁ֚
	dehi	שׁ֭
8	Above marks from the following group as they occur from right-to-left:	
	shalshelet	שׁ֓
	zaqef qatan	שׁ֕
	zaqef gadol	שׁ֔
	revia	שׁ֖
	zarqa	שׁ֗
	qarney para	שׁ֘
	gershayim	שׁ֞
	geresh muqdam	שׁ֝
	geresh	שׁ֜
	telisha gedola	שׁ֠
	iluy	שׁ֟
	qadma (*azla*)	שׁ֡
	ole	שׁ֫
	pazer	שׁ֡
	masora/number dot[3]	שׁ֨
	high punctum extra.[2]	שׁ֙
	masora circle[4]	שׁ֯
9	Postpositive above marks from the following group:	
	segolta	שׁ֒
	pashta	שׁ֡
	telisha qetana	שׁ֠
	zinor	שׁ֮

Notes:

1. See page 13 for specific information about *meteg* ordering.
2. See page 14 for specific information about *puncta extraordinaria*. Note that the high *punctum* is centered above the letter and above any other high marks; it should usually be ordered last of the marks in its group. Similarly the low *punctum* should usually be ordered after other below marks.
3. See page 15 for specific information about the masora/number dot.
4. Although it may visually be positioned slightly to the right of other above marks, the masora circle is usually ordered after any other marks in this group. See page 15 for specific information about *masora* circle handling.

Meteg handling

This section explains the various ways in which the meteg *mark can be applied and the expected results.*

The Hebrew mark *meteg* can appear in a variety of positions relative to other below marks, and all these are supported in the SBL Hebrew positioning lookups. However, not all are equally well supported in applications at the time of writing. This section explains how to encode sequences of consonant plus meteg with other marks to achieve different visual results, and shows how these will be ideally rendered.

The normal position of *meteg* relative to vowel marks is to the left, *e.g.* וַיִּמְצְאוּ (1 Kings 1:3). Such a sequence is encoded *consonant + vowel + meteg,* that is, as the marks are ordered from right to left.

Much less commonly, *meteg* can appear to the left of a cantillation mark, *e.g.* עֲבָדִֽים (Ex 20:2). Again, this sequence is encoded as the marks are ordered from right to left: *consonant + vowel + cantillation + meteg.*

When *meteg* is applied to a consonant bearing a *hataf* vowel (*hataf segol, hataf patah* or *hataf qamats*), the default font rendering is also for the meteg to be positioned to the left of the vowel, *e.g.* חֲיוֹת־אֱהִיֵֽה (Ps 50:21). However, it is common for this mark combination to be displayed with a medial *meteg* positioned between the two parts of the *hataf* vowel, *e.g.* אֱלֹהֵיכֶֽם (2 Chr 32:15). This is handled by the SBL Hebrew font, using an OpenType ligature substitution, when the Zero Width Joiner (ZWJ, U+200D) control character is inserted between the *hataf* vowel and the *meteg: consonant + hataf vowel* + ZWJ + *meteg.*

Finally, there are those instances in which *meteg* needs to be placed to the right of a vowel. These are relatively common on the first consonant of a word in the *Biblia Hebraica Stuttgartensia* text, *e.g.* תַּֽעֲשֶׂה־לְךָ֔ (Ex 20:4), but may also occur in mid-word, *e.g.* וְלַֽנַּעֲרָהּ (Deut 22:26). In such cases, *meteg* should be encoded before the vowel, *i.e.* as the marks are ordered from right to left: *consonant + meteg + vowel.* Note that this ordering should also be used in those rare cases where *meteg* occurs to the right of a *hataf* vowel, *e.g.* הֲלֹא־אַתָּה (Psa 85:7).

Note that because of the canonical combining class assigned by Unicode to the meteg character, it may be subject to reordering during normalisation (see pages 8–11). For instance, the example from Exodus 20—תַּֽעֲשֶׂה־לְךָ֔—would be reordered during normalisation such that the *meteg* shifts to the left of the *patah:* תַּֽעֲשֶׂה־לְךָ֔. If the relative position of meteg to a vowel is textually important, care must be taken to prevent reordering through automatic or manual insertion of the CGJ character, *e.g. bet + meteg* + CGJ + *patah.*

Puncta extraordinaria

This section explains the current recommendation for encoding the puncta extraordinadia, *illustrating how they should appear if rendered correctly.*

Puncta extraordinaria (extraordinary points) occur fifty-six times in the Old Testament text: fifty-three times above letters and three times below letters. The electronic edition of the *Biblia Hebraica Stuttgartensia* text notes that the function of these marks

> is not entirely clear, but it has variously been proposed that: a) the marks are merely emphasis and draw special attention to the theological implications of the word; b) the marks are early critical marks which indicate an omission or change that the scribes desired to make, but dared not; c) the marks represent drops of ink or even bits of dirt that were slavishly copied from one manuscript to the next; d) the marks indicate a special or unusual pronunciation of the word, or that the word should not be read at all; or e) some mixture of the above, on a case-by-case basis.

Biblia Hebraica Stuttgartensia : with Westminster Hebrew morphology. 1996 (electronic ed).

The most complex example of *puncta extraordinaria* is in Ps 27:13, in which *puncta* appear both above and below, and with other marks.

Note that, in accordance with the recommend mark ordering on page 12, the complex combination of *lamed* with above and below marks plus *puncta* in this example should be encoded *consonant + vowel + puncta below + mark above + puncta above*. Also note that because the *puncta extraordinaria* are positioning above and below the level of most other marks, it may be necessary to increase linespacing in those parts of the text where these marks occur.

The lower punctum was previously encoded in SBL Hebrew using the generic combining dot below (U+0323). *Documents produced with this earlier encoding will need to be updated.*

The SBL Hebrew font encodes these marks using the Hebrew 'upper dot' (U+05C4) and Hebrew 'lower dot' (U+05C5) characters; the latter is a new character in version 4.1 of the Unicode Standard.

Atnah hafukh and *qamats qatan*

This section explains two new characters in Unicode 4.1 and their implementation in this version of SBL Hebrew

The new *atnah hafukh* (U+05A2) character has been included in Unicode 4.1 for the benefit of those users who wish to encode an explicit distinction between this accent and *yerah ben yomo* (U+05AA). Many editions do not make this distinction, using the *yerah ben yomo* character and its glyph to represent both accents. Since this is a new character, with some disunification legacy issues (the form of the *yerah ben yomo* character used in many typefaces is actually that of the *atnah hafukh*), support in this version of SBL Hebrew is preliminary and subject to review: the character code is supported, but the glyph is identical to that used for *yerah ben yomo,*

The new *qamats qatan* (U+05C7) character makes it possible, if desired, to encode an explicit distinction between this vowel (short *qamats*) and the long *qamats* (U+05B8, *qamats gadol*). To visualise this distinction, following a convention adopted in some editions, the glyph for this new character is noticeably taller than that for *qamats gadol.*

Masora/Number dot

This section explains the use of the upper dot mark in Masoretic notes and for Hebrew numerals

Not to be confused with the high *puncta extraordinaria,* this high dot mark is used primarily in the context of Masoretic notes, as in this example from Num 32:24:

נוֹ־לָכֶם עָרִים לְטַפְּכֶם וּגְדֵרֹת לְצֹנַאֲכֶם
TEXT

בּ[14] . יוֹ מפק א[15]
MASORA

Like the low *puncta extraordinaria* the masora/number dot is not encoded in the Unicode Standard as a specifically Hebrew character, so a generic combining dot above (U+0307) is used.

For a complete table of Hebrew numerals, see R. Wonneberger, Understanding BHS, *1990.*

The same character is sometimes used to indicate a consonant used as a numeral in the traditional Hebrew numbering system: א=1, ב=2, ג=3 … י=10, כ=20, ל=30, and so on, with nine consecutive letters of the alphabet used for units, tens and hundreds up to 400 (some sources identify final letter forms for 500–900).

See, for example, Georges Ifrah, A universal history of numbers, *1998 (original French edition 1994).*

This numbering system is extended in some reference books to counting in thousands by doubling the dots: א̈=1000, ב̈=2000, ג̈=3000, etc.. The double dot is encoded using the generic combining diaeresis character (U+0308).

Masora circle handling

This section explains how to encode and control the positioning of the masora *circle.*

The circle indicating a Masoretic note can occur over a consonant, between consonants, over a *maqaf,* or over a word space. Ideally, it should be centred over a word, but there is no way to specify this in Unicode combining mark encoding or OpenType Layout. Manual care must be taken with the positioning. The SBL Hebrew font, by default, places the masora circle above the preceding consonant, *maqaf* or word space, as on these examples from the first chapter of Genesis:

מְרַחֶ֖פֶת … [9]אֶל־מָקוֹם … [11]עֵ֣ץ … רֵֽי[2]

The masora circle glyph is centered on a zero-width, which means that if the default positioning is inhibited it will automatically be placed between the character to which it is applied and the next. The positioning can be inhibited by inserting the ZWNJ control character between the masora circle and the consonant, *maqaf* or word space to which it is applied. In the examples below, also from Genesis 1, the character sequence is *consonant* + [*vowel/cantillation mark*(*s*)] + ZWNJ + *masora circle*.

לְמִקְוֵ֥ה … [10]
יַּמִּ֑ים … [28]וַיְבָרֶךְ

Note that, as shown in the last example from Genesis 1:28, the ZWNJ character can also be used to position a masora circle after a word break at the end of a line.

Holam male

This section explains the encoding of the vowel holam male, *distinct from* vav *with* holam haser.

Some texts—both manuscript and typographic—make a visual distinction in the position of the dot between a *vav haluma, i.e. vav* followed by a *holam haser* (וֹ), and the independent vowel *holam male* ( ). Some texts do not make this distinction, displaying both as , and some electronic documents may even encode both graphemes the same way, relying on the reader's expertise to distinguish them.

Earlier versions of the SBL Hebrew font used a mark+base ordering convention to distinguish *holam male* and *vav haluma,* but this has been deprecated because the *holam male* encoding was counter to basic Unicode rules about base+mark ordering. The correct encoding of this distinction has been the subject of much discussion in Unicode circles, and the standard is in the middle of a transition to a robust encoding distinction. The consensus is that the character sequence *vav* + *holam* (U+05B9) is the correct encoding for *holam male,* since this is far more common than the relatively rare *vav haluma.* This is implemented in version 1.12 of the SBL Hebrew font:

= *vav* + *holam* (*holam male*)

Documents using the earlier conventions of holam + vav = holam male / vav + holam = vav haluma *will need to be updated. Since such updates are complicated, we are no longer recommending any interim hacks to encode the distinction, but instead encouraging users to patiently await a robust and standard solution.*

Unfortunately, at the time of writing there is no standard means to encode a distinct *vav haluma.* A new combining mark character, HEBREW POINT HOLAM HASER FOR VAV, has been proposed and is approaching balloting, but is not available yet. If approved and encoded, this will provide a robust encoding distinction for those users who wish to differentiate *holam male* and *vav haluma* as illustrated in the fourth and fifth words of this example from Genesis 14:3:

וַיֹּאמֶר קַיִן אֶל־יְהוָה גָּדוֹל עֲוֹנִי מִנְּשֹׂא׃

Nun hafukha

This section explains a mechanism for displaying the 'inverted nun' glyph.

Nun hafukha—'inverted *nun*'—is not a letter, but a form of special punctuation that occurs only a handful times in the Hebrew Bible (see Numbers 10:35–36). Hebrew manuscripts show a number of different forms for this character, depending on the script style, only one of which corresponds to an inverted *nun* letter. This character is encoded in Unicode as the new character U+05C6.

׆ = *nun hafukha*

In pointed texts, the inverted *nun* carries a dot above it. For this, the masora/number dot character (U+0307) should be used:

׆̇ = *nun hafukha* + *masora/number dot*

Support in non-Windows systems & applications

This section explains the current state of support for SBL Hebrew on Macintosh & Linux operating systems.

The term 'font support' can refer to a number of different things, from installability to text display to complex glyph rendering, so when wondering whether a particular operating system or application (or particular combination of OS and application) supports use of a font it is necessary to understand the limitations that can apply.

Macintosh. The Mac OS X operating system can natively install and render the SBL Hebrew 'data-fork' format Windows TrueType font. Older versions of the Mac operating system cannot natively install this format, and need a Mac-specific 'resource-fork' format font file. At present, no resource-fork version of the SBL Hebrew font is available, since there are so few Unicode enabled applications for older Mac operating systems. Note that some applications, particularly new versions of much Adobe software (on both Mac and Windows), can install fonts, including SBL Hebrew, directly in their own font folders and use their own rendering technology, bypassing system font handling; some of these applications may provide support on older versions of the Mac operating system.

The latest version of OS X, Tiger, implements some system level support for basic OpenType Layout features, converting them on the fly to Apple's own AAT format features. However, this does not include support for complex script layout, so the all-important mark attachment is not supported at the system level. This means that applications relying on standard system calls for text processing will not be able to handle Biblical Hebrew text using SBL Hebrew, even though they may support Unicode text encoding and basic right-to-left layout.

Applications that use their own text layout engines, designed to work with OpenType fonts, offer the most hope for Macintosh users. At the time of writing, none of these applications offer the same high level of Biblical Hebrew rendering as Microsoft Office 2003 on Windows, but some are gradually getting close.

See Appendix C on page 22 for web link information on Mellel and other applications mentioned in this section,

Mellel is a word processing application for OS X that employs OpenType Layout natively using its own text engine. Biblical Hebrew test results from Mellel are encouraging, with almost all tested letter+mark sequences rendering correctly. One problem in the current release is that there is no way to turn off visual display of control characters, so if you insert *e.g.* the Zero Width Joiner character in a sequence Mellel will display a small mark above the letters. [In Microsoft Word on Windows, display of control characters is off by default, but can be turned on to facilitate editing; this behaviour is preferred.]

XeTeX is a typesetting system that marries Donald Knuth's TeX system with Unicode text encoding and OpenType Layout. Only a

small number of users appear to be using this for Biblical Hebrew, but the examples they have submitted are impressive and indicate that XeTeX provides some of the best support for Hebrew outside of the Windows version of Word and other Office applications. XeTeX makes use of the ICU layout library discussed below.

Linux. As with many other aspects of open source computing, the level of support for Biblical Hebrew on Linux depends very much on specific distribution, library and application configurations. This section of the manual is necessarily short simply because we have limited experience in this area, and have only recently begun to explore it.

Users of the SBL Hebrew font who are interested in Hebrew rendering on Linux systems are encouraged to investigate the ICU and Pango, two code libraries that provide support for Unicode text processing including OpenType Layout support. Tests of current version of Pango, compiled into the Firefox web browser, show correct positioning of single accent to single vowel combinations, but errors in second accent and accent-to-accent relative positioning. Results from applications using ICU seem better.

Linux-based users of the SBL Hebrew font are particularly encouraged to join the MSN user community, and to share tips and tricks for working with Biblical Hebrew text on this platform.

Conclusion

As should be clear from this manual, the SBL Hebrew font is on the 'cutting edge' of text processing and layout technology, and much software still needs to catch up to the font in order to perfectly render all aspects of Biblical Hebrew texts. That said, in the seventeen months since the last version of this manual was written, there have been encouraging advances, especially on the Mac and Linux systems.

During development of the font, key software developers such as the Microsoft development lead responsible for the Uniscribe Hebrew engine have consulted on the best way to encode and shape Biblical Hebrew and have provided beta versions of upcoming software releases for testing. This consultation has enabled us to make a font that outperforms all previous Hebrew text rendering, and which will enable Biblical scholars to create, edit and exchange documents between the increasing number of systems and applications that support Unicode text encoding and OpenType Layout.

Appendix A
End User License Agreement

This license agreement explains the rights and responsibilities that you accept when you install the SBL Hebrew font on your computer. Please take a few minutes to review this.

1. The digitally encoded machine readable font software for producing the typefaces licensed to you is the property of Tiro Typeworks. It is licensed to you for use under the terms of this end user license agreement. If you have any questions about this license agreement, or have a need to use the font software in a way not covered by this agreement, please write to license@tiro.com.

2. You may use this font software free of charge for all non-commercial purposes. If you wish to obtain a license for commercial use of this font software, please contact the Society of Biblical Literature at sblexec@sbl-site.org, or write to license@tiro.com. Fees for commercial licenses are at the individual discretion of the Society of Biblical Literature and Tiro Typeworks.

3. You may redistribute this font software free of charge as long as the software is unmodified, all copyright and trademark notices are intact, and the font contains or is accompanied by a copy of this license agreement. You may not charge any fee for the distribution of this font software or alter the terms of this license agreement.

4. You may decompile and modify this font software for non-commercial and personal use by you as an individual or internal use within your organisation. Tiro Typeworks maintains copyright to all derivative fonts in any format. You may not delete, edit or add to copyright, trademark or license information in the font. You may not change the embedding bit. You may not redistribute any modified version of the font software, either free of charge or for a fee. Copies of modified fonts should be submitted to Tiro Typeworks (license@tiro.com) and to the Society of Biblical Literature (sblexec@sbl-site.org), along with any relevant documentation. Tiro Typeworks reserves the right to incorporate any such changes into its own fonts.

5. You may embed the font software in non-commercial electronic documents, including but not limited to web pages and e-books. Font embedding must respect the embedding bit in the font, which must not be changed. The embedding bit for this font software is set to 'Editable Embedding', meaning that documents containing this font software may be viewed, printed and edited, but the embedded font may not be installed on the recipient user's system.

6. All other rights are reserved by Tiro Typeworks, except as otherwise designated in contract between Tiro Typeworks and the Society of Biblical Literature.

7. Neither Tiro Typeworks not the Society of Biblical Literature warrant the performance or results you may obtain by using this font software. Excepting any warranty, condition, representation or term that cannot or may not be excluded or limited by law applicable to you in your jurisdiction, Tiro Typeworks and the Society of Biblical Literature make no warranties, conditions, representations, or terms (express or implied whether by statute, common law, custom, usage or otherwise) as to any matter including, without limitation, noninfringement of third party rights, merchantability, integration, satisfactory quality, or fitness for any particular purpose.

8. Neither Tiro Typeworks nor the Society of Biblical Literature accept any liability for injury, death, financial loss or damage to person or property (including computer hardware, software or data) resulting from the use of this font software.

9. The act of installing this font software on any computer system constitutes acceptance of the terms of this license agreement, without exception.

Appendix B
Custom combining classes.

As explained on pages 8–11, Unicode normalisation may break Biblical Hebrew text by reordering marks that should not be reordered. This appendix provides alternate combining classes to use in custom normalisation routines. **Nota bene:** *these alternate combining classes are outside of any recognised standard, and text produced using custom normalisations may still be subject to other normalisations in software beyond the author's control* (e.g. *web browsers*). *This list is provided purely as a suggestion, and no guarantee is made that it will be supported as is in software developed by SBL, Tiro Typeworks, or any of their project partners.*

See the discussion on pages 8–11 for more information about normalisation issues, and the Unicode Standard for details of standard normalisation and combining classes. Below are suggested canonical combining classes to use in custom normalisation routines for Biblical Hebrew text. As in standard normalisation, the expectation is that only marks in different combining classes will be reordered during normalisation; marks with the same combining class value should not be reordered. The lower the combining class value, the closer to the base character (*e.g.* Hebrew consonant) the mark should be ordered. In this table, the recommended combining class is provided first, then the existing Unicode combining class in grey, and then the Unicode codepoint and a descriptive name.

10	24	U+05C1	Point Shin Dot
11	25	U+05C2	Point Sin Dot
21	21	U+05BC	Point Dagesh or Mapiq
23	23	U+05BF	Point Rafe
27	19	U+05B9	Point Holam
220	220	U+05C5	Lower Punctum
220	220	U+0591	Accent Atnah
220	220	U+05A2	Accent Atnah Hafukh
220	220	U+0596	Accent Tipeha
220	220	U+059B	Accent Tevir
220	220	U+05A3	Accent Munah
220	220	U+05A4	Accent Mahapakh
220	220	U+05A5	Accent Merkha
220	220	U+05A6	Accent Merkha Kefula
220	220	U+05A7	Accent Darga
220	220	U+05AA	Accent Yerah Ben Yomo
220	10	U+05B0	Point Sheva
220	11	U+05B1	Point Hataf Segol
220	12	U+05B2	Point Hataf Patah
220	13	U+05B3	Point Hataf Qamats
220	14	U+05B4	Point Hiriq
220	15	U+05B5	Point Tsere
220	16	U+05B6	Point Segol
220	17	U+05B7	Point Patah
220	18	U+05B8	Point Qamats
220	18	U+05C7	Point Qamats Qatan
220	20	U+05BB	Point Qubuts
220	22	U+05BD	Point Meteg
222	222	U+059A	Accent Yetiv
222	222	U+05AD	Accent Dehi

230	230	U+05C4	Upper Punctum
230	230	U+0593	Accent Shalshelet
230	230	U+0594	Accent Zaqef Qatan
230	230	U+0595	Accent Zaqef Gadol
230	230	U+0597	Accent Revia
230	230	U+0598	Accent Zarqa
230	230	U+059F	Accent Qarney Para
230	230	U+059E	Accent Gershayim
230	230	U+059D	Accent Geresh Muqdam
230	230	U+059C	Accent Geresh
230	230	U+0592	Accent Segolta
230	230	U+05A0	Accent Telisha Gedola
230	230	U+05AC	Accent Iluy
230	230	U+05A8	Accent Qadma
230	230	U+05AB	Accent Ole
230	230	U+05AF	Mark Masora Circle
230	230	U+05A1	Accent Pazer
230	230	U+0307	Mark Number/Masora Dot
232	228	U+05AE	Accent Zinor
232	230	U+05A9	Accent Telisha Qetana
232	230	U+0599	Accent Pashta

The makers of the SBL Hebrew font would like to thank the individuals and organisations who participated in ad hoc discussions to determine an appropriate mark ordering for normalised Biblical Hebrew, especially Peter Constable (SIL International) and Eli Evans (Libronix/Logos). Other contributors included Joan Wardell (SIL International), Patrick Durusau (SBL), Ralph Hancock, Paul Nelson (Microsoft), Bob Pritchett (Libronix/Logos), & Kent Richards (SBL).

Appendix C
Resource links

This appendix provides URLs for font and document downloads and online resources from the Society of Biblical Literature and other parties.

`http://www.sbl-site.org/Resources/default.aspx`
The latest version of the SBL Hebrew font, this manual, the consonant + mark(s) test document, and other information relating specifically to SBL Hebrew; downloadable Windows keyboard drivers for input of Biblical Hebrew text.

`http://www.sbl-site.org/`
Society of Biblical Literature website.

`http://www.tiro.com/resources/hebrew`
Tiro Typeworks resources for Biblical Hebrew, including normalisation example documents.

`http://www.unicode.org/`
The Unicode Consortium website. Includes latest version of Unicode Standard and resources.

`http://www.microsoft.com/typography/specs/default.htm`
OpenType specification and Microsoft Hebrew font specification.

`http://www.microsoft.com/typography/developers/opentype/default.htm`
'Windows Glyph Processing' article by John Hudson; an introduction to OpenType font and Unicode script processing on Windows.

`http://www.redlers.com/mellel.html`
Mellel is a powerful multilingual word processor for Mac OS X. It's handling for Biblical Hebrew is not yet perfect, but it is very good.

`http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=xetex`
Download site for the open source XeTeX page layout application for Mac OS X.

`http://www.winsoft.fr/`
Winsoft make the Middle East versions of Adobe software, including InDesign ME, which was used to typeset this manual.

`http://www-306.ibm.com/software/globalization/icu/index.jsp`
International Components for Unicode: an open source set of libraries for Unicode support, software internationalization and globalization. Provides excellent OpenType Layout support.

`http://www.pango.org/`
An open-source framework for Unicode text layout and rendering.