A System for Typesetting Biblical and Modern Hebrew with Omega and TeX
Version 2.0, January 2003

Alan Hoenig
ahoenig@suffolk.lib.ny.us
Makor:
A System for Typesetting Biblical and Modern Hebrew with Omega and \TeX

Alan Hoenig
Department of Mathematics
John Jay College, City University of New York
ahoening@suffolk.lib.ny.us

January 17, 2003

Contents

1 Why use \textit{Makor}? 4
  Earlier versions of \textit{Makor} 5
2 Getting and installing Makor 5
  Installation 5
  Testing the installation 7
  Other files 7
3 Why use \textit{\TeX}? 8
  The \textit{\TeX} life cycle: a brief introduction 9
  Creating a simple Makor document with Lambda 9
  Creating a simple Makor document with Omega 10
4 Some Makor conventions 11
  Consonants 11
  Vowels 12
  More about \texttt{8} 13
  Punctuation, dashes, foreign sounds, and so on 13
  Numbers 14
  Pattern breaking 14
  Indentation and \texttt{white space around a paragraph} 14
  Adjustments 15
  Fine tuning 15
  Cantorial trope 16
  Examples; irony 18
5 Fonts 18
  All the characters in a font 18
  Font selection 19
  A Makor reference card 21
  Missing bold fonts 21
Old Hebrew .................................................. 24
Additional fonts .......................................... 24
Working with TrueType ............................... 24

6 Conventions
Variations on the הָיְי convention ......................... 26
Non-Makor conventions: ArabTEX, Biblia Hebraica Stuttgartensia .... 27
Yiddish and Ladino .................................. 28
Makor "dialects" ...................................... 29
Archaic systems: Old Hebrew, Palestinian Babylonian vocalization .... 31
Standard conventions; rolling your own... ............ 32

7 Hebrew alignments and tables
Protection ............................................... 34
Suppressing reversal .................................. 35

8 Various and sundry
A trick or two ........................................ 36
Processing this document ............................ 38
Caveat! .................................................. 39

List of Figures
1 Learning to read Hebrew ............................ 3
2 A portion of the Book of Esther .................. 6
3 Fully vocalized Makor output (see file vocal.tex)........ 9
4 A page of Talmud ...................................... 12
5 Hebrew intermingled with English .............. 16
6  הָיְי keyboard conventions for cantorial trope .... 17
7 A font table for Ezra ............................... 20
8 The Siloam water tunnel inscription ........... 21
9 Available Makor fonts ............................. 22
10 A Makor reference card ......................... 23
11 A portion of Genesis using Hadassah types .... 25
12 Yiddish input conventions ....................... 28
13 The Yiddish Jabberwock! ......................... 30
14 A German dialect of Makor ....................... 31
15 Archaic Hebrew vowel systems ................. 31
16 Standard Makor environments ................. 32
17 A bit of Esther: trope, but no vowels .......... 33
18 Sample Hebrew alignments ....................... 35
19 Using a Hebrew table to display a verb conjugation .... 36
20 The Siloam water tunnel inscription: enhanced display .... 37
21 A portion of Ecclesiastes ...................... 40
Abstract

This manual describes the *Makor* (📚) system for typesetting Hebrew by means of Omega, an extension of the TeX typesetting language. *Makor* is appropriate for authors using English-language keyboards for entering Hebrew text in a convenient manner. *Makor* lets you enter vowels and cantorial diacritics, selects final letters by itself as appropriate, and has special conventions for entering numerals and tables. The software plus fonts is freely available from the Web.

All of the Hebrew text in this manual was produced using *Makor*. I produced all other displays using programs that are part of the TeX or Omega distribution. I urge all potential authors to flip through to see if *Makor* would be suitable for you.

Figure 1: Learning to read Hebrew. The simple appearance of this primer belies the difficulties involved. Interested readers should carefully go over the source for this figure (*primer.tex*).
If you need to include Hebrew text in a document—especially in combination with English or other left-to-right (LTR) language—you know what horrors you’re in for. Text entry itself can be challenging, and vowel placement adds to your worries. Makor attempts to eliminate these issues. So if you need to include Hebrew within a report or article, here’s why you should use Makor:

- It’s easy to enter consonants and vowels into a document.
- Makor automatically decides if a final form for a letter is necessary.
- Makor takes care to position vowel symbols properly with respect to different letters.
- There are easy ways of finagling these (and other) aspects—altering the placement of a vowel, forcing or suppressing the final form of a letter, and so on.
- Authors can decide whether to include vowels or not.
- Authors can enter cantorial diacritics (trope) into the text.
- Makor also supports Yiddish with a separate input convention and special Yiddish characters.
- Alternative typographic conventions are supported, such as \( \checkmark \) instead of \( \checkmark \) and \( \checkmark \) for \( \checkmark \).
- Fonts include the special characters (\( \checkmark \) and so on) so that it’s possible to typeset the Hebrew Bible with Makor.
- Numbers are entered normally, and Makor takes care to typeset them properly within the Hebrew (that is, the numbers are LTR even though the surrounding text is RTL).
- Makor knows the traditional Hebrew numbering system (see, for example, figure 2), and can number Biblical verses automatically.
- Makor understands the conventions of ArabTEX, so you can process ArabTEX Hebrew documents in Makor.
- Makor understands the conventions of BHS, so you can process Biblia Hebraica Stuttgartensia in Makor.
- One of Makor’s fonts allows scholars to typeset Old Hebrew.
- Authors and scholars can typeset using the archaic Palestinian or Babylonian vowel systems.
When you revise the document—either by changing your text or altering layout parameters (such as the width of a column, or whatever)—these changes automatically propagate into your text.

- Layouts of arbitrary complexity are possible.
- All of the versatility that's part of $\TeX$ and of Omega is always available to the author using this system.
- ...And much, much more!

Earlier versions of $\text{םָאָרָו}$

An earlier version of $\TeX$ worked with any extended $\TeX$ that provided for right-to-left (RTL) typesetting. The current version demands the use of Omega, and as a result, the current version of $\text{מקָּרוּ}$ is far more robust, and contains several additional capabilities.

I deprecate this former version and urge those users to switch to the current version as quickly as possible.

2 Getting and installing Makor

You can always download the latest version of this software from the Web; the appropriate site is www.ctan.org. I mention this site from time to time throughout this document, and it goes by the name of 'ctan'. It contains lots of other software suitable for use with $\TeX$, $\text{לַעַמָּאָרָו}$, and Omega, including several other packages for typesetting Hebrew, such as Arab$\TeX$ (which has a Hebrew module), Hebrew $\text{לַעַמָּאָרָו}$, and a worthy package called cjhebrew. Please make sure that you download all the files as one large compressed file, as this makes installation particularly easy. Depending upon your method of download, this file might be makor.zip, makor.tar.gz, makor.tar.Z, or something similar. For our discussion, I'll refer to it as makor.zip, though it might have one of these other names.

Incidentally, the full address for the latest version of $\text{מקָּרוּ}$ is

www.ctan.org/tex-archive/language/hebrew/makor

Installation

I take it as given that $\TeX$—including Omega, Lambda, and their friends—has been properly installed on your system. If not, consult your local $\TeX$ wizard, or local guides available at ctan. Another source is the $\TeX$ Users Group (tug; www.tug.org).

Make sure you know where the 'root' of your local $\TeX$ directory is. Most likely, it has a name like '../texmf/'. Henceforth, I will call this the <texmf-root> directory.
LIBER ESTHER

A portion of the Book of Esther, showing both cantorial and vocal diacritics. Notice too that can handle both spoken (q’re) and written (k’tab) variants of text. The verse numbering was done automatically. See the file esther.tex for additional details.
Place yourself in this `<texmf-root>` directory, and uncompress `makor.zip`, making sure that you preserve directory structure (this should be the default behavior).

Next, move to the `<texmf-root>/tex/makor2` directory. Place the contents of the file `psfonts.mkr2` into your file `psfonts.map`, which is located someplace like

`<texmf-root>/dvips/base`

Now, update the filename database (the procedure to do this is a key part of your \TeX implementation), and that's it!

Testing the installation

Within this same `<texmf-root>/tex/makor2` directory are some useful files. Enter the commands

`omega mkrotest`

and

`lambda mkrltest`

at the terminal. This exercises 'plain' Omega and Lambda, which is Omega+\IF\TeX. You should get two `.dvi` output files, which you can preview or print after you enter these commands:

`odvips mkrotest`

`odvips mkrltest`

I use GhostView for previewing and printing PostScript files. If you prefer to work with `.pdf` output, substitute the command `dvipdfm` for `odvips`.

Other files

Most of the Makor2 package consists of font-related files. A few other useful utilities are present, however. For example, you can use the file `fonttbl.tex` to print all the characters of a standard computer font (not more than 256 characters). For further discussion, please see below, page 18.

A somewhat similar file is `fontgrid.tex`, which prints out superlarge images of selected font characters superimposed on a grid. This file is useful for creating the specialized font metrics for a Hebrew font, and directions for its use appear within this file itself.

Also present is `refcard.tex`, which you can use to print a two-page summary of Makor2 and its conventions. See below, page 21, for further discussion.

The `makefonts` directory contains all the files that I used to create the virtual files that are part of Makor2. I include them for the sake of completeness, but strongly recommend that all but experts give this material a wide berth.
The remaining eps and doc directories contain the files you'll need to recreate this manual for yourself. See page 38 below for additional information.

3 Why use \TeX?

The point of \TeX{} (and of Omega) is to create documents whose appearance is of sufficient quality to qualify for the most demanding publishing requirements. \TeX{} automatically generates indexes, tables of contents, and automatically inserts other typesetting minutiae, such as ligatures, where they belong. \TeX{} was originally created to handle English-language typesetting, and it's easy to generalize this to work with any Latin-based language. Omega includes all the capabilities of \TeX{}, and adds a mechanism for handling non-Latin typesetting in a reasonably natural way. Although \texttt{Makor} works \texttt{only} with Omega, I will frequently refer to '\TeX{}' or '\LaTeX{}' or 'Omega' even though I only mean 'Omega' in these contexts.

The Omega or \TeX{} life cycle is quite different from the usual Windows or Macintosh \textit{gestalt}. If you're a typical computer user, you expect as a matter of almost divine right to type stuff at the keyboard and see the document composed and formatted before your eyes, onscreen. But there are reasons why this method may not work to your advantage; I can mention only a few here.

Optimal line breaking depends, for example, not only on what you've typed so far, but also on the words in the remainder of the paragraph. In extreme examples, the very last word of the paragraph can determine the linebreak of the first line. How then can any other program claim to give you world-class formatting if it composes paragraphs on the fly?

I expect my program to decide for me whether a medial or final form of a letter is needed. Yet, in a what-you-see-is-what-you-get program, it's difficult for the program to know when I've reached the end of a word. By the time you type a space or punctuation, a medial letter has already appeared on the terminal. \textit{WYSIWYG} programs never backtrack to revise themselves.

Finally, although the menu selection system which accompanies typical commercial software offers legion choices, no set of menus can encompass all the formatting choices that an author might make. Therefore, anyone serious about the look of the written word must think seriously about the kind of programming interface—so that you can implement any format you need—that \TeX{} provides.

As with any sophisticated computer tool, advanced \TeX{} use can be challenging and rewarding. Nevertheless, it's easy to make \TeX{} (and \textit{Makor}) do simple and straightforward things. Further information about \TeX{} appears below.
The \TeX{} life cycle: a brief introduction

Creating a document—article, report, book, whatever—with \TeX{} is quite different from using a commercial word processor or publishing program. You begin by preparing a strictly \texttt{ascii} text file containing the text of your report, say, amongst which are interspersed various formatting commands.

Next, you run through the \TeX{} program, as you run a file containing the text of a computer program through a program compiler. In \TeX{}'s case, the output is a file with the extension \texttt{dvi}, a device-independent file which contains very general instructions for rendering letters and other graphic elements on the printed page.

The \texttt{dvi} file is not something you can deal with directly, so we need one final step. Some subsequent program has to render the file. For example, I typically run my stuff through the excellent program \texttt{dvips}, which converts the \texttt{.dvi} file to a PostScript (\texttt{.ps}) file; actually, you need the Omega variant \texttt{odvips}.

In practice, it takes no time to implement these tasks, and they can be combined into batch or shell scripts to make the typesetting process as easy as possible.

Before using \texttt{Makor}, you'll need to install it (as with any software program). See above, section 2, page 5, for installation instructions and discussion.

Creating a simple \texttt{Makor} document with Lambda

For people who are curious, or perhaps just a bit unsure, here's how you use \texttt{Makor} to create a simple Lambda document incorporating both Hebrew and English, where 'Lambda' is the Omega version of 'LATEX'.

Using a text editor, create a \texttt{Makor} \LaTeX{} test file called \texttt{mkrltest.tex} (or something similar) that contains the following purely \texttt{ascii} text.

\begin{verbatim}
\documentclass{article}
\usepackage{makor2}

\begin{document}
Hello, world! \[\text{\texttt{\textbackslash{ash\text-	extbackslash{aul}}O\textbackslash{m}, \texttt{\text\textbackslash{Olaum}}!}\] Goodbye, world!
\end{document}
\end{verbatim}
\end{document}

For those familiar with \LaTeX, this document is largely familiar. The document section of the file contains text delimited by Hebrew typesetting switches \{ and \}. The ascii within these toggles will get translated and typeset into fully vocalized Hebrew by Omega.

To enable this process, issue the command

\texttt{lambda mkrltest}

from the pos-prompt or its equivalent. If Omega and Makor are both properly installed, and if you haven’t made any errors when you created your test file, then Omega will tell you that the file \texttt{mkrltest.dvi} is now present in the current directory. The typeset contents of this file cannot directly be viewed or printed without converting it to PostScript or Acrobat pdf format.

I prefer PostScript output, so I issue the next command in this process

\texttt{odvips mkrltest}

to create the file \texttt{mkrltest.ps}, which I can now view or print with a current version of GhostView. (All these programs, along with Omega and Lambda itself, should be present as a result of a proper installation of \TeX and Omega. You can also use GhostView to convert your .ps into the Acrobat .pdf format.) It will look something like this: Hello, world! \HlUm,rUlHm! Goodbye, world!

Any file you create for typesetting with Makor + Lambda should follow this schematic.

\textit{Creating a simple Makor document with Omega}

For those of you who prefer to use plain Omega, the only change you need to make to the above instructions has to do with the form of the input Makor test file \texttt{mkrotest.tex}. In this case, the file has the simple form:

\begin{verbatim}
\input makor2
Hello, world! \[sh`aulOn, ‘Olaum!\] Goodbye, world!
\bye
\end{verbatim}

and you run the command ‘omega mkrotest’ followed by ‘odvips mkrotest’.
If you know Hebrew, perhaps you will agree that the input required to produce the short display on the previous page is a good match for the Hebrew itself. Nothing special was done to get the final form for the א’s—Makor chose them automatically. Experienced \TeX-users know that short selection above relied upon default values of the system, and these can be changed many ways to alter the typeset appearance. In this document, we will explore the defaults built in to Makor, beginning with the conventions you need to produce Hebrew letters.

You get letters, vowels, and cantorial diacritics using the lowercase English consonants and vowels, and the special characters ' , `, ^ , _, | , * , and the uppercase 0. Of course, punctuation and numerals are used in the usual ways as needed.

All the conventions we now discuss are summarized below in Figure 10 on page 23. (This table includes alternative keyboard conventions as well.)

Consonants

Where possible, you get the Hebrew consonant that you expect from the corresponding English input. Thus, you type b to get ב, f to get פ, and so on. Some Hebrew consonants don’t correspond to single Latin letters, so we use reasonable compromises. For example, type sh to get של, and type ts and ch to get צ and ג. We use ' and ` to get ש and ר; these are long-standing conventions that apply to all Semitic languages. We use a prepended period to specify relatively rare consonants; thus, we get ק and ח from .ק and .ח.

Almost all Hebrew letters can occur in a dotted form. We pronounce some of these dotted forms differently, and their input is straightforward. We’ve seen above how to get ב already. We get ס and פ via т and п. At other times, we need the dot for grammatical purposes, even though this may not change the pronunciation. One way to get this dotted form is to append an asterisk, which visually approximates the appearance of a dot. Thus ה* produces כ. But dots are also added for a grammatical reason—when they correspond to a doubled consonant. We can use this interpretation to make it easy to “dot” a letter—simply double it. So we get מ and צ from yy and dd, and so on.

Makor makes other conventions possible as well. For example, we use the circumflex ^ to get the upperdot on the י. Thus, י comes from either מ or צ and we get י from שhא.

At times, you may want to suppress the automatic generation of final letters, or conversely, force the typesetting of a final form. To this end, use the ` and _ characters to force and suppress these forms. Thus, גמ_\' yields the abbreviation for ‘gemara’: גמ, and you get the silly מָן from p\'p. (Mnemonic: just as the ` is suspended in mid-air, so too the final form of a letter is followed by ‘air’, the following word space.)

Sometimes, perhaps to fool Makor that you’ve reached the end of a word, you may
Thus $7\%$ or $8\%$, or its synonym $-\%$, to add a word space, and to then remove it. Thus \texttt{\textbackslash t\textbackslash t\textbackslash a\textbackslash} is almost the same as \texttt{\textbar t\textbackslash t\textbackslash a\textbackslash}. 

**Vowels**

*Makor* takes great care with placement of vowels. Thus, although a vowel is centered more-or-less geometrically on a letter like $\texttt{ב}$ (ב), and so on), they are placed—properly—all the way to the right for $\texttt{ר}$ (ר) and so on.

One or more English vowels map to a Hebrew vowel, and the vowel follows the consonant to which it belongs. Thus, to get ̀, type ̀̀. For some vowels, you need two letters, such as ̀ or ̀̀. Some vowels have alternative encodings, and you
choose the one that seems most reasonable for you. For example, \textae is an alternative to \textoe, and you might use this, because it seems somehow more fitting to get \textbneiy yi'srau'ael.

We use + as an alternative to au for the qamats vowel, because + looks a bit like this vowel. We use the colon to help generate the bataf-vowels: : from h:e. The vertical bar helps yield the meteg-vowels: | yields \textw, and so on.

Sharp-eyed readers will have noted that the right single quote is used to generate the schwa vowel, as well as the letter aleph. \textMakor takes care to distinguish between the two cases, and does a good job at doing the right thing. In case you need to enforce one or the other of the cases, again use ` and _ . Thus, ` always yields whereas ` always typesets \textK. (Mnemonic: both the ` symbol and the letter aleph are above the line, whereas the underscore and the shewa are at or below it.)

The sole use of uppercase in the \textMakor keyboard conventions occurs in vowel representations. Whereas o generates the defective cholam, you need O for the full cholam U.

More about \textK

In certain conditions, a holamdot followed by \textK needs proper placement. This is done automatically; hence \textK from bo'. In other situations, the holamdot should be on the preceding consonant. Thus, to get \textbo we type bo\textbackslash', using the pattern breaker (see below). In these two words, look carefully at the positioning of the holamdot.

Punctuation, dashes, foreign sounds, and so on

By and large, Israeli punctuation matches that of the rest of the world. One difference from American grammar is the frequent occurrence of the sequence `?! to indicate some level of amazement incredulity. Just for fun, I have included in my Hebrew fonts the 'interrobang' symbol ¡, which you get by typing either ?? or ??!. Perhaps this symbol will now sweep the world of Hebrew typesetting by storm¡. See below if you want to typeset the usual sequence.

Type ** to get the 'newsheqel' symbol ¥

In the usual way, we type --- and to get em-dashes. If you type --, you get ~, not quite a hyphen in the usual sense. If the font contains a hyphen, you get it by typing =.

In modern Hebrew, it's customary to use single quotes \text ' to indicate a foreign, non-Israeli sound. Thus, t\textbackslash'a and g\textbackslash'oon generate \textcheck and \textjungle (pronounced 'check' and 'jungle'). \textMakor takes care of the case that these foreign sounds occur at a word ending. Hence, rabiynOvits' typesets \textra'igail. Note well—\text' does not work; type only \text' in this case.
Numbers

The convention in Hebrew (and Arabic too!) is that 'Arabic' numerals are to be read in the usual left-to-right manner, despite the fact that surrounding Hebrew text is read in the opposite way. Makor understands numerals, and for simple cases, takes care of the directional switch automatically.

More complicated cases—involving decimal points and whatnot—will be treated more appriately in an upcoming version of this software.

Pattern breaking

Omega works by examining patterns in the input and replacing various patterns by Hebrew characters according to rules that are part of the Makor system. Occasionally, we need to break them. For example, we may want the sequence ?! or ** instead of ? or מ.

To this end, we use the command \/. This command sort of looks like a wedge facing down, and you can think of it as a wedge that breaks up the pattern. Thus, to get ו type t/א.

Indentation and white space around a paragraph

Because of the inherent left-to-rightness of Omega, you have to be the tiniest bit alert about white space and paragraph indentation. For example, if you type

```
\[laukhaen chakoo liy n'um ydwd l'yDm qoomiy lau'ad kiy mish''pau tiy l'eqe sof g0ym l'qauv' tsiy mam'laikh0s lish''pokh'
':alayhem za''miy kol ch:arOn 'apiy kiy b''eish' qin''ausiy t|''+kael k+1-h+'[++rets.\]
```

you get:

 kole mehulaynu meroz la'ad khamel yadim bimshesh peloh yis'hem kol pelohe m'alohot shelof.

This is no good for two reasons: (1) the first line is indented improperly; and (2) the last line is ‘filled’ with white space on the right side.¹

Simple Makor commands remedy both deficiencies. First off, you can suppress the improper left indentation with the usual \noindent command. If you terminate the Hebrew paragraph with \HPAR, you get proper paragraph filling, so that

```
\noindent[laukhaen chakoo liy ... k+1-h+'[++rets:\HPAR\]
```

yields

¹Incidentally, this text, which is drawn from the book of Zephania, 3:8 (that is, זפניה 3:8), is a good test of Hebrew typesetting. It contains all of the letters of the Hebrew alphabet, including the final forms in addition. Thus, this verse is the Hebrew equivalent of “the quick brown fox ...”
However, to enforce a Hebrew indentation on the right, use the command \HINDENT. Also, the command \CENTERLASTLINE is a variant of \HPAR, and will (as you would imagine) center the last line of the paragraph (something I can’t quite figure how to do in standard \TeX typesetting). Thus,

\noindent\[
\HINDENT laukhaen ... k+1-h+I+rets: \CENTERLASTLINE\]

yields

לַעַ֣הַנֶּךָ שֹׁאֱרְדֲרֵ֥ו לַאֹ֣ה קָוָ֣ה לְ֣הָרֵ֔ד כָּנַ֣שְׁפָּ֔שֵׁל לֵ֙אהֵ֣כָה נִצָּ֔בִית מְ֥לָכָ֖ה לַאֹ֣ה

which is common convention in some Hebrew liturgical typesetting.

Adjustments

From time to time, a vowel may not appear exactly where you’d like it. \TeX, of course, contains a plethora of commands which allow for precise placement of typographic elements, but these may be awkward to use in the context of setting Hebrew. For this reason, you can embed adjustment commands—finagles, if you will—to fix things up. All these fine tunings apply only to the next consonant+vowel pair; usual metrics are restored thereafter.

The first kind of adjustment will simply move the next vowel (and this vowel only!) a fraction of the character width to the left or right. You enter this adjustment by surrounding the decimal fraction with a pair of underscores.

For example, we normally type \texttt{rau} to get \texttt{ך}, in which the qamats is properly centered under the tail of \texttt{ך}. For some reason, you may wish to place the vowel at the center of the character instead. If you type \texttt{_.5\_raurau}, you get \texttt{ךך}, which also emphasizes that the adjustment only lasts for one character.

Fine tuning

We use the expression fine tuning to refer to a more careful adjustment, where you want to carefully specify position changes in both horizontal and vertical directions. If you surround a pair of valid, comma-separated \TeX dimensions by underscores, then \texttt{Makor} uses this set of dimensions as an offset to the original position. For example, the silly \texttt{_.3.5pt,.3.5pt\_rau} gives rise to \texttt{ך}. Note the convention of signs used in fine tunings.
Rabbinic Hebrew (רל) does not differ greatly from Biblical Hebrew (בכ) in its inflection of the noun, although the neutralization of final mem and nun means that the masculine plural is often, as in Aramaic, רל. Apart from the more frequent use of the archaic feminine suffix רל, as in רל,… with the feminine ‘dumb woman’, רל also employs the suffixes רל. and רל. for example רל,… in Aramaic and רל…. ‘servitude’. רל developed distinctive feminine plural suffixes in רל,… (Babylonian) or רל…. (Palestinian), for example רל,… from bath-houses and רל…. as in רל,… ‘kingdoms’ for רל,…, for nouns ending in רל…. in the singular. Masculine plural forms sometimes differ from those that would be expected, or are normally found, in רל, for example, רל,… from רל… ‘damage’, רל,… from רל… ‘ox’, רל,… from רל… ‘market’, רל,… from רל… ‘side’, רל,… from רל… ‘half’, and רל,… from רל… ‘envoy’. The same is true of feminine nouns, for example רל,… from רל… ‘letter (of alphabet)’, רל,… from רל… ‘covenant (without plural in רל)’, and רל,… from רל… ‘mother’.

Some masculine nouns take the feminine plural suffix רל…, for example, רל,… from רל…. ‘favour’, רל,… from רל…. ‘rule’, רל,… from רל…. ‘baby’, רל,… from רל…. ‘army’, רל,… from רל…. ‘city’, and רל,… from רל…. ‘water’. Similarly, there are some feminine nouns which take the masculine plural suffix רל… רל…. from רל…. ‘dove’, רל…. from רל…. ‘ant’, and רל…. from רל…. ‘egg’, for example. Occasionally, both types of plural are evident, as with רל…. רל…. from רל…. ‘day’ or רל…. רל…. from רל…. ‘year’, with each form having a slightly different shade of meaning and the ‘feminine’ variant only used with suffixes. In רל we sometimes find plurals of nouns only attested in the singular in רל, for example רל,… from רל…. ‘limb’, רל,… from רל…. ‘grass’, and רל,… from רל…. ‘daily sacrifice’. Likewise, there are singular forms of nouns only attested in the plural in רל, for example רל,… from רל…. ‘coral-wood’, רל,… from רל…. ‘egg’, and רל,… from רל…. ‘onion’. The dual is used more than in רל, with existing forms retained and new ones created, for example רל,… ‘scissors’ and רל,… ‘meanwhile’. (1993: A. S’aenz-Badillos, A History of the Hebrew Language, Cambridge University Press, pp. 188-89.)

**Figure 5**: Hebrew intermingled with English.

**Cantorial trope**

I found it tough to think of a convincing entry convention for the cantorial diacritics. Each of these symbols has a Unicode name, though, and *Makor* accepts this name, surrounded by square brackets, to stand for the trope. Thus, we get ר from ר[etnahta]. You may feel this is a lot of typing, but you are free to define command shortcuts. Thus, after

```
def\8{[etnahta]}
```

we get the same result by typing \8\8. Of course, you can (and should!) define command names that are more meaningful to you.
If you don’t like the names that Makor uses for these accents, complain to the Unicode naming committee. Figure 6 lists these names and the associated trope symbols.

Metafont is a sibling program of TeX that can be used for high quality font design. I used this utility to create the trope characters, which may explain their lack of quality. If any reader knows of a better trope font that could be used instead, I’d be grateful for the tip.

<table>
<thead>
<tr>
<th>שָׁלְשֶׁל</th>
<th>תֵּלְהַגַּדְּלָ</th>
<th>שָׁלְשֶׁל</th>
<th>תֵּלְהַגַּדְּלָ</th>
</tr>
</thead>
<tbody>
<tr>
<td>סָגוֹל</td>
<td>פָּזֶר</td>
<td>סָגוֹל</td>
<td>פָּזֶר</td>
</tr>
<tr>
<td>שַׁלְשֶׁל</td>
<td>מַעֲנָה</td>
<td>שַׁלְשֶׁל</td>
<td>מַעֲנָה</td>
</tr>
<tr>
<td>צַעְּפְּגַּרְּטָן</td>
<td>מַחֲפֵקָ</td>
<td>צַעְּפְּגַּרְּטָן</td>
<td>מַחֲפֵکַ</td>
</tr>
<tr>
<td>צַעְּפְּגָדֹ</td>
<td>מֶרֶךְ</td>
<td>צַעְּפְּגָדֹ</td>
<td>מֶרֶךְ</td>
</tr>
<tr>
<td>תְּפֵהַ</td>
<td>מֶרֶךְ מַחֲפֵקַ</td>
<td>תְּפֵהַ</td>
<td>מֶרֶךְ מַחֲפֵקַ</td>
</tr>
<tr>
<td>רֵבִיעָ</td>
<td>דָּגָ</td>
<td>רֵבִיעָ</td>
<td>דָּגָ</td>
</tr>
<tr>
<td>זְרָקָ</td>
<td>קָדָמְ</td>
<td>זְרָקָ</td>
<td>קָדָמְ</td>
</tr>
<tr>
<td>פַּשְׁחַ</td>
<td>הֶלְקֲחַ</td>
<td>פַּשְׁחַ</td>
<td>הֶלְקֲחַ</td>
</tr>
<tr>
<td>יֵטִיעָ</td>
<td>יֵרַהַבְנָיָוּ</td>
<td>יֵטִיעָ</td>
<td>יֵרַהַבְנָיָוּ</td>
</tr>
<tr>
<td>טְבִיר</td>
<td>אוּל</td>
<td>טְבִיר</td>
<td>אוּל</td>
</tr>
<tr>
<td>גֶּרֶשְ</td>
<td>יָלוּ</td>
<td>גֶּרֶשְ</td>
<td>יָלוּ</td>
</tr>
<tr>
<td>גֶּרֶשְ-הַמַּעְדָ</td>
<td>דֶּהָ</td>
<td>גֶּרֶשְ-הַמַּעְדָ</td>
<td>דֶּהָ</td>
</tr>
<tr>
<td>גֶּרֶשְ-הָזָיָ</td>
<td>צִנּוּר</td>
<td>גֶּרֶשְ-הָזָיָ</td>
<td>צִנּוּר</td>
</tr>
<tr>
<td>קָרַנְיַ</td>
<td>מָסֵּר</td>
<td>קָרַנְיַ</td>
<td>מָסֵּר</td>
</tr>
</tbody>
</table>

Figure 6: Keyboard conventions for cantorial trope.

Makor makes it possible to fine tune these accents as well. If a single accent appears above a letter, the syntax is as before, except you enclose the correcting factor by circumflexes instead of underscores. Thus,

1°r[salsheshelet]

yields 1° and

3pt,1.5pt°r[salsheshelet]

typesets 1°; normally, you get ° from r[salsheshelet].

Under a letter, it’s possible to have two accents, and you can specify a semicolon-delimited second pair of coordinates to indicate an additional amount of offset between
the accents. Here’s an example to illustrate the meaning of this gobbledygook. Normally, if you type \texttt{zau[etnahta]} you get \textsuperscript{†}. Possessed by a spirit of anarchy, you could enter \texttt{.1pt,1pt;3pt,2pt_zau[etnahta]}; this produces \textsuperscript{†}. That is, the dimensions in the first pair offsets the vowel-trope pair, while the second pair adjusts the spacing between the two diacritics.

I find myself rarely needing any of these adjustments. If you find yourself entering more than one every few pages, it may mean that the \textit{font} as a whole needs fine tuning. This is not difficult, but the procedure is out of place in this document. Please contact me if you feel there is a problem with one of the \textit{Makor} fonts.

\textit{Examples; irony}

The sources for all the examples in this document are part of the \textit{Makor} distribution. Please refer to them frequently for guidance in using the \textit{Makor} conventions.

[Ironically, the name in English for this software—\textit{Makor}—does not follow the above rules. If it did, it would be \texttt{mauqOr}! Sorry for this inconsistency, but it's too late now.]

5 \textbf{Fonts}

Everybody loves fonts, and the fonts that are part of \textit{Makor} provide a feast for the eyes. Dealing with fonts is a two-pronged operation, involving selection of a particular font and access to all the characters of the font. Incidentally, all of \textit{Makor}'s fonts are free (or some variant thereof), and I am continually seeking additional \textit{high quality} fonts to add to the package. I'd be grateful if any readers with suggestions for additional fonts could get in touch with me. (Thanks!)

\textit{All the characters in a font}

The vast majority of a the characters in a font are accessible by means of \textit{Makor}'s keyboard conventions. As we've seen, if you want \texttt{ê}, you type \texttt{l}, and so on. Nevertheless, it's possible that a \textit{font} contains additional characters that may not be so readily accessible. For that reason, the \textit{Makor} package includes a file called \texttt{fonttbl.tex}, which allows you to print out all characters in a file. Within the first half-dozen lines, there is a line like

\texttt{\def\myfont{ezra2 at 12pt}}

and you should replace the font name and font size as appropriate. A quick list of the fonts that are part of \textit{Makor} forms figure 9. (Note that there are nine gray fonts, named \texttt{ezrag12} through \texttt{ezrag92}; \texttt{g} = gray.) The \texttt{fonttbl} document doesn't depend upon Omega, so you can compile it with \TeX; that is, first issue the command

\texttt{\texttt{tex fonttbl}}

\textbf{18}
and then convert it to PostScript or pdf via either of the commands
dvips fonttbl
or
dvipdfm fonttbl

Figure 7 displays the font table for the standard Ezra font. The first line of the table, the first sixteen characters of the font, are reserved for future use. Every font contains some special, perhaps even oddball, characters, and Makor sticks them into the second row. The last row is empty, and the remaining slots are filled with the characters of a Makor font.

Makor provides a \PrintChar character to access some of the more esoteric characters, but this command takes the character number as the argument, so it's useful to print out this table and retain it for future reference. For example, since the reflected nun appears in position 136, you could print it via \PrintChar{136}. Of course, this is somewhat clumsy. A better approach would be to create a command
\def\reflectednun{\PrintChar{136}} %%% Omega (plain TeX) syntax
or
\newcommand\reflectednun{\PrintChar{136}} %%% Lambda (LaTeX) syntax

Font selection

Makor expects an author to select fonts one by one. In the absence of instructions to the contrary, the Ezra normal font is automatically selected for Hebrew typesetting; this is the font used up to this point in this manual.

Using a font involves two distinct instructions. The first simply makes the font visible to Makor. That is, it associates a name that you create with a particular font at a particular size. Thus, after
\hfontdef{RSH}{rashi}{10pt}
the name RSH conjures up—to Makor—a 10 pt version of Rashi script. (\hfontdef=Hebrew font definition.) To invoke this font, you need to include the command
\hfont{RSH}
by your text. (\hfont=use Hebrew font.) As with any scope command in any version of \TeX, such a command obeys \TeX's grouping laws. The \hfont should follow the \{ command.

Figure 9 displays a bit of all the currently available Makor fonts. The leftmost field in each line shows a sample of a font, while the middle field give the Makor name, which is the name you use in a \hfontdef or \hfont command. The final field gives a human name for the font, along with the design size of the sample. As you can see, there is a huge visual variation in fonts at the same size. Notice too that all modern fonts have Makor names ending in ‘2’, indicating their suitability for use with Makor2.
Figure 7: A font table for the default Ezra font.
Figure 8: The Siloam water tunnel inscription; see siloam.tex for details.

A Makor reference card

Figure 10 shows a portion of a reference card which summarizes all the Makor coding and usage conventions. You can generate and print a more extensive such card if you wish. To do so, run the file refcard.tex through Omega and odvips, and print it on a letter-size sheet or card. Before doing so, you may have to (or you may wish to) change some of the fonts used in the reference card. The initial part of this file suggests the means for doing this.

Missing bold fonts

If you look closely at figure 9, you'll notice that not every variant for every font is present. In particular, it's often difficult to create a decent bold-faced font. For that reason, Makor provides two related macros that might be useful in this regard. Following the discussion in The TeXbook pertaining to “poor man's bold,” Makor provides commands \pmb and \PMB to embolden their arguments. They work better for smaller and larger fonts respectively. So if you type

'vgdh \pmb{wzch.ty} khlmn.s

you get

אכזרת חחס קלימנות

These macros work by placing several copies of the text slightly offset from each other, so the results are a tad fuzzy or even fuzzy (I used \pmb and \PMB here), but it may be better than nothing. Note that it is not possible to typeset extended texts this way, since it is not possible to typeset text which breaks at lines with these macros. (If the look of this bold is sufficient for your needs, and you need extended passages in poor man's bold, you'll need to create a suitable virtual font for this purpose.)
The Hebrew alphabet in all of the currently available Makor fonts.
<table>
<thead>
<tr>
<th>Consonants</th>
<th>Star Convention</th>
<th>'Doubles'</th>
<th>Basic Vowels</th>
<th>More Vowels</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>^d</td>
<td>d&quot;</td>
<td>^h</td>
<td>y:a</td>
</tr>
<tr>
<td>s</td>
<td>^s</td>
<td>s&quot;</td>
<td>^w</td>
<td>j</td>
</tr>
<tr>
<td>t</td>
<td>^t</td>
<td>t&quot;</td>
<td>^y</td>
<td>n</td>
</tr>
<tr>
<td>y</td>
<td>^y</td>
<td>y&quot;</td>
<td>^v</td>
<td>i</td>
</tr>
<tr>
<td>r</td>
<td>^r</td>
<td>r&quot;</td>
<td>^z</td>
<td>l</td>
</tr>
<tr>
<td>n</td>
<td>^n</td>
<td>n&quot;</td>
<td>^h</td>
<td>am</td>
</tr>
<tr>
<td>z</td>
<td>^z</td>
<td>z&quot;</td>
<td>^w</td>
<td>m</td>
</tr>
<tr>
<td>c</td>
<td>^c</td>
<td>c&quot;</td>
<td>^y</td>
<td>y</td>
</tr>
<tr>
<td>m</td>
<td>^m</td>
<td>m&quot;</td>
<td>^v</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>^f</td>
<td>f&quot;</td>
<td>^z</td>
<td>p</td>
</tr>
<tr>
<td>p</td>
<td>^p</td>
<td>p&quot;</td>
<td>^w</td>
<td>m</td>
</tr>
<tr>
<td>w</td>
<td>^w</td>
<td>w&quot;</td>
<td>^y</td>
<td>c</td>
</tr>
<tr>
<td>l</td>
<td>^l</td>
<td>l&quot;</td>
<td>^v</td>
<td>e</td>
</tr>
<tr>
<td>k</td>
<td>^k</td>
<td>k&quot;</td>
<td>^z</td>
<td>h</td>
</tr>
<tr>
<td>j</td>
<td>^j</td>
<td>j&quot;</td>
<td>^w</td>
<td>s</td>
</tr>
<tr>
<td>th</td>
<td>^th</td>
<td>th&quot;</td>
<td>^y</td>
<td>t</td>
</tr>
<tr>
<td>n</td>
<td>^n</td>
<td>n&quot;</td>
<td>^v</td>
<td>m</td>
</tr>
</tbody>
</table>

**Commands**

- \{ start Heb. typesetting
- \} finish Hebrew
- \( pattern breaker
- \( suppress indentation
- \textbf{INDENT} enforces a Hebrew indentation
- \textbf{\H\par} terminates a Hebrew paragraph
- \textbf{\H\centerlastline} centers last line of paragraph
- \textbf{\H\unskip} unskip
- \textbf{\H\sy} synonym for unskip
- \textbf{\H\font} defines a Hebrew font for use
- \textbf{\H\font} uses the named font
- \textbf{\PrintChar{242}} identifies a character by slot, and prints it
- \textbf{\MakorEnvironment{MKR}} selects typesetting environment

**Adjusting Vowel Placement**

- `-4..rau` moves the au vowel 40% of the width of the glyph starting at the left and moving to the right.
- `-3.5pt,-3.5pt_rau` displaces the au vowel 1.5 pt to the left of where it should be and 1.5 pt up from where it should be.
- `-9y[shalahelet]` puts the upper cantorial accent 90% of the way to the right of the left edge.
- `-3pt,1.5pt` displaces the upper accent by 3 pt to the left and 1.5 pt up.
- `-1pt,1pt,-1pt_rau[etsahta]` to displace the pair of vowels and the increase the offset between them.

**Figure 10:** Portion of a Makor reference card. The full card fits on two sides of a letter-size sheet.

You get the Hebrew character in the left column by typing the keystrokes in the right.
Old Hebrew

The Old Hebrew font is far sparser than other fonts, so you should be more careful about how you use it. It is another Metafont hack by the current author, and readers knowing of a better font should get in touch with me.

Figure 8 shows the famous Siloam inscription rendered in Makor’s Old Hebrew glyphs.

Additional fonts

It’s a straightforward (albeit a trifle tedious and time-consuming) to add additional fonts to the Makor system. As an example, I adapted the beautiful MasterFont Hadassah fonts (available from Studio Rosenberg, 159 Yigal Alon St., Tel Aviv 67443, Israel; 159 창강로 320-159, 서울특별시) to the Makor system. Unfortunately, as they are proprietary, they cannot be distributed with this package, but see figure 11 to see how they look.

Working with TrueType

These MasterFont fonts arrived in TrueType format, which are not readily understood by all variants of \TeX. (However, \TeX’s sibling pdf\TeX does, for example.) Here are some tips for working with TrueType fonts. This information may be of use, as the TrueType format is becoming more and more common.

In many cases, I use the superb ttf\_edit utility to massage the fonts into ones better suitable for use by Makor. (This program’s author, Richard Kinch at kinch@truetex.com can help you get hold of this program.) Then, I used the invaluable TrueTypetoType42 utility by Thomas Barton (available from the ctan archives) to create PostScript files that dvips recognizes. Another alternative is the ttf2mf utility, a means for converting TrueType into Metatfont source files, which is available from various archives.

Finally, you should know about the wonderful pfaedit utility, which is a general program for working with fonts in a wide variety of formats. With this program, you can manipulate individual characters as well, like commercial programs such as Fontographer. pfaedit is freely available from various locations on the Web. Please note, though, that this is primarily a Unix/Linux-type program, and though it has been ported to other operating systems, such as Win32, you’ll need to download additional software to get it working. (In my case, I had to download Cygwin, a Unix-like terminal for Windows, and XFree86, a free implementation of X-Windows, to get pfaedit to work on my Win95 computer.)
In the beginning of God’s creating the heavens and the earth — when the earth was wilderness and void, with darkness over the surface of the deep, and the breath of God was hovering upon the surface of the waters — God said, “Let there be light,” and there was light. God saw that the light was good, and God separated between the light and the darkness. God called the light: “Day,” and to the darkness He called: “Night.” And there was evening and there was morning, one day.

Figure 11: A portion of Genesis. The Hebrew is MasterFont Hadassah. See genesis.tex for details.
We use the term 'convention' to refer to system by which authors direct Makor to do their typesetting. So far, we’ve only encountered the standard Makor convention, by which, for example, you type š to get š, you type shah-e[etnahta] to get א, and so on. However, there are other conventions available, and this section introduces them to you.

The friendly term 'conventions' is related to the less friendly piece of jargon 'opc list', which may be familiar to you if you’ve glanced at the Omega documentation. This is a list which tells Omega which rules to apply to the input. Makor creates these lists dynamically based on your wishes. We’ll describe the more common conventions before concluding this section with a brief discussion on how to create your own special Makor environment.

We specify a new Makor environment by means of the command

\MakorEnvironment{...}

or its synonym

\MakorConvention{...}

where you replace the ellipsis by a valid environment name. By default, the standard Makor environment, named MKR, is in effect.

Variations on the convention

Several variations on the MKR environment (set of conventions) are already part of the Makor system. The first concerns an alternative to the vowel placement in the final ק character. Normally, you see ק but if I type this as

\[\MakorEnvironment{MKRalt}kh’ khau\]

I get ק instead. To get ק ק ק ק ק ק ק I type something like

\[\MakorEnvironment{MKRalt}l’khau laekh’ \MakorEnvironment{MKR}l’khau laekh’\]

Similar comments apply to a ‘bent lamed’ convention which must have originally been adopted to save space. There is a ‘bent lamed’ MKR variant. I get ק ק ק ק ק ק ק by typing

\[\MakorEnvironment{MKRbl}l’khau laekh’ \MakorEnvironment{MKR}l’khau laekh’\]

(Here, bl=bent lamed).

As if to emphasize the lesser value of vowels over consonants, most Hebrew typesetting omits the vowels. Exceptions seem to be made for children’s books and texts.
foreign words and names (which no one would know for sure how to pronounce), poetry, and liturgical texts. It’s easy to omit vowels in a Makor document—simply omit the vowels! Of course, this leads to problems in proofreading. It’s much easier to read

lakol z’muun w’‘aes l’khaul–chaefets tachas hashsh‘+m+yim:

instead of lkhl zmn w’s lkhl-chfts achs hahym: when you want to typeset לְלָל instead of לְל: Consequently, there is a MKR variant to do that. This is the environment MKRnv (nv=no vowels). I got the preceding vowelless phrase by typing

\[\text{\textbackslash\textbackslash MakorEnvironment{MKRnv} lakol \ldots hashsh‘+m+yim:}\]

Perhaps systems like Makor which make it as easy to typeset vowels (and other diacritics) as consonants may cause the wider use of vocalized reading material.

Non-Makor conventions: Arab\TeX, Biblia Hebraica Stuttgartensia

A few important Hebrew keyboard conventions exist prior to the development of Makor, and experiments with two of these show that it is feasible to adapt these conventions to Makor.

Some years ago, Klaus Lagally developed an ingenious system for typesetting Arabic with \TeX, and shortly afterwards he extended this system to handle Hebrew. (Both scripts are alike in being written rtl and in treating vowels as optional diacritics.) His system is quite robust, but authors using this system are limited to special fonts that are part of the system. By this time, there is a substantial corpus of input files adhering to this system, so I thought it would be a good idea to adapt the Arab\TeX-Hebrew conventions for use with Makor.

After typing \textbackslash MakorConvention\{Lagally\}, input such as .b.irU.a_h _t0vhAh typesets as בּוֹרֳא_ח נאָ_ת0וּאָח (this is b’roo_ach .t0vauh in the standard Makor convention). The Arab\TeX package, with its manual, is widely available. Most readers can browse their local ctan site (www.ctan.org) for this material.

The case of bhs (Biblia Hebraica Stuttgartensia) is more interesting. This text is a careful rendering of the Hebrew Bible, containing standard and cantorial diacritics, oddly shaped characters, oddly vocalized characters, oddly sized characters, and other miscellanea as per the most ancient extant manuscripts. The bhs encoding is more a work-in-progress than a finished project. For some reason, although large chunks of this file compile perfectly in Makor, I cannot run the entire bhs file as a whole through Makor without generating incomprehensible errors. Reader input/advice/suggestions are very welcome.

Anyway, an ASCII file with this bhs information is publicly available, but it adheres to its own system of transcription. For example, the first bit of Esther looks like

Figure 12: Yiddish input conventions. The last three lines summarize the conventions by which a leading N is applied automatically to certain vowels, or by which it is suppressed.

It is possible, of course, to typeset Yiddish with Makor by simply pretending that it's some kind of funky Hebrew, and enter it using the usual Makor conventions (together with some definitions to access those Yiddish characters that typically do not appear in Hebrew). But this is silly. It's easy enough to provide a Yiddish environment whose conventions are summarized in figure 12.

It's worth pointing out that some sounds are duplicated in Yiddish. For example, both v and .v produce different letters which have the same sound. Several of these duplicates have been taken directly from Hebrew, probably so certain printed Yiddish words...
look like their Hebrew antecedents. These letters are the ones in figure 12 whose input is preceded by a period.

Certain sounds need a leading ` when they begin a word; Makor takes care of this automatically, as you see in the bottom of figure 12. In these last three lines, the input text o¿ is given a leading depending on its position in the word. A leading underscore, as in _o¿ suppresses the leading.

Figure 13 provides an example of Yiddish typesetting, displaying the Yiddish translation of “Jabberwocky” by Raphael Finkel; check file jabberwocky.tex for details of the typesetting, but the conventions for the first stanza are

s'iz brilik geven, di shlikhtinke toves
hobn gevirt un gevimlt in vobn:
gants mimzish geven di borogoves;
di mome-ret hot oysgegrobn.

which seems to mirror the actual pronunciation pretty well (although I am certainly no Yiddish maven!).

I believe that the Makor fonts are all “Ladino-ready.” That is, they contain all glyphs you might need to typeset Ladino text. Alas, I have not been able to locate an unambiguous table of sounds for a Ladino alphabet. If any reader can provide this, I will be happy to create a Ladino convention for Makor. This applies to any other script or dialect using Hebrew characters.

*Makor “dialects”*

One great advantage of Omega is that it’s so easy to customize your input scheme. We’ve seen examples of that above, and here’s one final one.

I tried hard to make the default input mechanism for Makor be as intuitive as possible, but, of course, intuition is a relative thing. It’s entirely reasonable for me, as a speaker of American English, to use the inputs ts and z for the letters tsaddik and zayin. But it’s not so intuitive, say, for a German user of Makor. Germans represent this pair of sounds by z and s respectively, and we might design a German ‘dialect’ of Makor to reflect this different linguistic perspective. Most input conventions are the same for a German speaker as for an English speaker, but there are some differences, so it makes sense to contemplate a Makor convention in which Germans, say, can enter Hebrew text in a way that makes sense for them. To this end, I’ve created the MKRdeu environment (deu=deutsch). Most of the input conventions remain the same as for the standard set of conventions, and the few differences appear in figure 14, and examples of its use appear at the bottom of this figure.

I make no claim to any expertise in German, and German readers may feel that my input choices are poor. Therefore, I welcome suggestions for improvement, as well as suggestions for other Makor dialects.
Jabberwocky
Lewis Carroll (1871)
Translated into Yiddish by Raphael Finkel

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

{Yiddish translation}

He took his vorpal sword in hand:
Long time the manxome foe he sought—
So he rested he by the Tumtum tree,
And stood awhile in thought.

{Yiddish translation}

He left it dead, and with its head
He went galumphing back.

{Yiddish translation}

He chortled in his joy.

{Yiddish translation}

He took his vorpal sword in hand:
Long time the manxome foe he sought—
So he rested he by the Tumtum tree,
And stood awhile in thought.

Figure 13: The Yiddish Jabberwock! See jabberwock.tex for details.
Archaic systems: Old Hebrew; Palestinian Babylonian vocalization

As we mentioned above, you get Old Hebrew (sometimes called Paleo-Hebrew) as a font change. Thus, after something like

```latex
\hfontdef{OH}{oldheb}{14pt}
...\[\hfont{OH}'vgdh wzch.tyk lmn.s' ftsqrsth\]
```

you get a set using these alternative vowel systems, Makor provides an experimental way for you to do that. Figure 15 shows an example of these systems.

![Archaic Hebrew vowel systems](archaic.png)

Figure 15: Archaic Hebrew vowel systems. See archaic.tex for typesetting details.

Makor performs this typesetting within the standard MIRR environment, but I’ve created special Ezra fonts containing the archaic vowels in place of the usual Tiberian diacritics. For example, I could define fonts such as

```latex
\hfontdef{Pal}{ezrapal2}{14pt}
\hfontdef{Bab}{ezrabab2}{14pt}
```

which I invoke after a call to a special \ArchaicVowels command. The examples in figure 15 was keyed according to the following schematic.
\def\PAL{\ArchaicVowels \hfont{Pal}}
\\[
MKR Standard Makor conventions; the default.
MKRbl Like MKR, except uses bent lameds.
Lagally Conventions of Klaus Lagally’s Arab TeX.
BHS Conventions of Biblia Hebraica Stuttgartensia.
BHSbl Like BHS, but with bent lameds.
MKRnv Like MKR, but no vowels or diacritics are typeset.
BHSnv Like BHS, but no vowels are typeset.
MKRalt Like MKR, but non-standard placement of vowels in some terminal letters is used.
MKRdeu a German (deutsche) variation of the standard MKR environment.

Figure 16: Standard Makor environments.

Standard conventions; rolling your own...

Makor sets up several conventions (environments) for typesetting in a variety of contexts. Figure 16 lists these, most of which have already been mentioned. Any of the items in the left column of this figure can appear as argument to the \MakorEnvironment command. For example, to typeset Yiddish, your input file would contain the following symbolism:

\MakorEnvironment{YID}\[\hfont{...} \]

(as usual, the ellipsis ... represents your text).

You may feel you need to create a variant of one of these. Makor will cooperate, provided you adhere to the following procedure.

First of all, Makor creates a temporary file to contain the instructions needed to customize your set of conventions, and you need to provide a name for this file. If you provide the name mystuff (for example), Makor creates the temporary file mystuff.tmp. You use this name in two places. This name is the argument of a command that Makor uses to begin the process. Next, you set the various parameters on or off that you want to adjust; see the file makor2.tex for details, but don’t change anything in it! After this, issue a command which tells Makor to begin the one or more new conventions, not forgetting to provide a name for this new convention. Finally, Makor needs to slurp up this information which has been written to the temporary file.

This description has been a bit too vague to be of use, so we present an example. Assume we need an environment that suppresses vowels but does print the cantorial trope at the same time that the Biblia Hebraica Stuttgartensia input rules are observed. The following statements should appear in your source file:

\StartSomeOCPLists{mystuff}
\Vowelloff \BHSon
\MakeMakorEnvironment{TRPVW}
Inhale Environment Knowledge{mystuff}
and you use this new environment in this way.

Makor Environment{TRPNV}\BIBLE[...]

Figure 17 shows the same selection as figure 2, but with all vowels stripped away. You rarely (if ever?) see Hebrew texts printed like this, but perhaps this may prove useful for people studying the chanting of the Hebrew sacred texts.

Figure 17: Selection from Esther, vowelless but with trope. See trpnv.tex to see how this was done.

7 Hebrew alignments and tables

I'd like to describe a macro—a variation of plain \TeX's \halign—that makes it possible to type Hebrew columns left to right and have them typeset right to left. I used \halign as a model rather than \LaTeX's \tabular because I've always found that \halign is more versatile. Caveat! This macro works by writing certain information to a temporary file; it is this operation that allows Makor to reverse the order of columns. However, as a result, strange things are apt to happen if you go too wild in your table macros. Some of this strangeness will be discussed below. Always be alert to the fact that you may have to reorganize or your tabular data or set up the table in a different way to get it to typeset. I invite motivated readers to re-do these macros more robustly.

Simple tables, such as the one in figure 6, require no special treatment. The table in figure 19 is a slightly different story. Makor provides a

\HEBALIGN
alignment macro for producing tables like this. Although \HEBALIGN is no way near as robust or versatile as \halign (the plain \TeX antecedent of \LaTeX's \texttt{\textbackslash tabular} environment), I hope it will be of use. Basically, we seek to enter enter columns and column data from left to right and have it typeset in right to left format.

As with all \TeX environments, \HEBALIGN requires a template followed by any number of row data. Both the rows and the templates are arguments of a \texttt{\CR} macro, which \texttt{\textbackslash precedes} the template or row data. As in standard \TeX, the tabbing symbol & separates each column. \textit{Unlike} standard \TeX, there must be the same number of &'s on each line. (Otherwise, the data will line up in a manner you won’t have anticipated.) In particular, \TeX’s neat && convention for the template row can \texttt{not} be used in \HEBALIGN. Essentially, any formatting commands can appear in the template specification. However, you cannot use the # tabbing symbol; there is a category code conflict. Use the Makor macro \H instead. Thus, a simple table could be typed as

```
\tabskip=1.5pc
\HEBALIGN{\CR{\hfil\H&\H\hfil}}% end of template
  \CR{'&baeyth}% end of first row
  \CR{gimel&d}% end of second row
} % end of \HEBALIGN
```

and which typesets as the top sample in figure 18. This could also have been coded as

```
\HEBALIGN{\CR{\hfil\H&\H\hfil}}% end of template
  \CR{'&baeyth\}}% end of first row
  \CR{gimel&d}}% end of second row
} % end of \HEBALIGN
```

There's also a \HEBNOALIGN macro, which works like the usual \texttt{\noalign} command. To get the second sample in figure 18, type

```
\HEBALIGN{\CR{\hfil\H&\H\hfil}}
 \HEBNOALIGN{\vskip2pt\hrule\vskip2pt}
 \HEBNOALIGN{\vskip2pt\hrule\vskip2pt}
 \HEBNOALIGN{\vskip2pt\hrule\vskip2pt}
}
```

\textit{Protection}

Owing to the aforementioned conflicts in category codes, you need to protect a command which is not primitive. For this you use the Makor command

\texttt{\textbackslash MPROTECT}

inside the \texttt{\textbackslash definition} for \texttt{\textbackslash myprotects}, which Makor examines. Thus, if we type

```
\def\myprotects{\MPROTECT\multispan\MPROTECT\bf}
\HEBALIGN{\CR{\H&\H&\H&\hfil[\H]\&[\H]\hfil}}% end of template
```

34
\HEBNOALIGN{\vskip2pt\hrule\vskip2pt}
\CR{\multispan3:\hfil Silly\hfil:&\multispan2\hfil\[sh`aulOm!\]\hfil}
\HEBNOALIGN{\vskip2pt\hrule\vskip2pt}
\CR{\[1\]&\[2\]&\[3\]&\&baeyth}\% end of first row
\HEBNOALIGN{\vskip2pt\hrule\vskip2pt}
\CR{1&2&3&\omit\bf Gimel&d}\% end of second row
\HEBNOALIGN{\vskip2pt\hrule\vskip2pt}
\HEBALIGN{\CR{\H&\H&\H&\hfil\[\H\]\&\[\H\]\hfil}} % end of \HEBALIGN

you get the third specimen in figure 18; this table requires some comments: (1) the value of \tabskip is 6.0pt; (2) numerals taken from the prevailing Hebrew font may well differ from those taken from the prevailing Roman font; and (3) I needed to \MPROTECT both \multispan and \bf, the latter because the \text{T\HEBALIGN{\vskip2pt\hrule\vskip2pt}}/NFSS assigns a rather different expansion than does plain \text{T\HEBALIGN{\vskip2pt\hrule\vskip2pt}}X. When in doubt, you can always \MPROTECT a command; it should do no harm in any case.

**Suppressing reversal**

If necessary, you can suppress the reversal process of several columns. Do this by enclosing the data (not the template) in curly brackets, and these brackets will certainly span several columns. For example, if we make the simple change

\tabskip=2pc\def\myprotects{\MPROTECT\multispan\MPROTECT\bf}
\HEBALIGN{\CR{\H&\H&\H&\hfil\[\H\]\&\[\H\]\hfil}}
\HEBNOALIGN{\vskip2pt\hrule\vskip2pt}

\textit{Figure 18}: Sample Hebrew alignments.
to the code for the previous table, we get the final table in figure 18.

<table>
<thead>
<tr>
<th></th>
<th>ובד</th>
<th>יבד</th>
<th>נבד</th>
<th>שמ</th>
<th>פבד</th>
</tr>
</thead>
<tbody>
<tr>
<td>הבט</td>
<td>מהלך</td>
<td>מהלך</td>
<td>מהלך</td>
<td>מהלך</td>
<td>מהלך</td>
</tr>
<tr>
<td>הבט</td>
<td>מהלך</td>
<td>מהלך</td>
<td>מהלך</td>
<td>מהלך</td>
<td>מהלך</td>
</tr>
<tr>
<td>הבט</td>
<td>מהלך</td>
<td>מהלך</td>
<td>מהלך</td>
<td>מהלך</td>
<td>Mahalch</td>
</tr>
</tbody>
</table>

Figure 19: Using a Hebrew table to display a verb conjugation.

Figure 19 puts most of this stuff together. You can check out the file hitpael.tex to examine the innermost details of its production. When you make this examination, note carefully the shenanigans with \tabskip, which look odd but aren’t. Remember, the template line is reversed in the typesetting process.

8 VARIOUS AND SUNDRY

In this section appear various topics that need to be in a manual, yet don’t quite fit into the other sections. These include some bread-and-butter like installation as well as some reference to various tricks.

A trick or two

Sometimes it pays to peruse the guts of the makor2 style file as well as some of the .otp files. For example, if you do this you will note that Makor surrounds glyphs with
TEX macros that are responsible for the placement of these characters and symbols on the page. You can—carefully—redefine these macros in aid of the occasional special effect. For example, figure 8 displays the old Hebrew letters that are part of the inscription of the old Siloam water tunnel in Jerusalem. It's interesting, of course, but most modern readers of Hebrew will want to know what each character represents without having to constantly flip back and forth to a font table. This inscription contains only consonants, each of which is placed in position by a macro called \PutCon (con=consonant). We carefully redefine this macro so it places the modern Hebrew equivalent underneath the old Hebrew consonant. Here's the redefined macro:

\def\PutCon#1{\oalign{#1\crcr \hidewidth\hfont{HDEFAULTFONT}#1\hidewidth \crcr}}

(see The \TeX{}book for further details). The enhanced display appears in figure 20.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{siloamalt.png}
\caption{An enhanced display of the Siloam water tunnel inscription; see siloamalt.tex for details.}
\end{figure}
Processing this document

I used Omega1.15 to process this document. Although this is not the latest release (as of January, 2003), it appears to be far more robust than version 1.23, at least according to reports on the Omega list.

It proved impossible to compile this entire document in Lambda (that is, the \LaTeX counterpart to Omega) without overflowing Omega's memory resources. As a result, I've included most of the figures by (1) creating separate files for these figures; (2) compiling them separately and creating .eps files of the output; and (3) including the .eps images in this file by means of the utility dvips (actually, the Omega-aware variant, odvips). I've included all files with the Makor package that you'll need to typeset this manual (with one exception described below). Thus, to typeset this manual, after having properly installed Makor (see above), you type these commands

\begin{verbatim}
lambda mkr2man
odvips mkr2man
\end{verbatim}

to generate mkr2man.ps, which you can view or convert to .pdf format. I encourage you to examine the underlying .tex file for this manual as an aid to learning how to use Makor. To better understand how a figure was typeset, refer to the appropriate source file. For example, to see how figure 1 was created, have a look at primer.tex.

As a result of having to include umpteen of these .eps files, the resulting PostScript and .pdf files are quite swollen. My abject apologies to all.

The only exception to my former statement—that all relevant files are part of the Makor distribution—has to do with fonts. This document is typeset using the Monotype Janson font family, a proprietary set of fonts. If you do not have these fonts, then comment out statements like\[\input zmtmjn\]
or\[\usepackage{zmtmjn}\]
which appear at the beginning of any of the .tex files. With these statements commented out, Lambda and Omega will use the standard \TeX fonts in the compilation.

If you do have these fonts installed, then do nothing. I have included the style files zmtmjn.tex, zmtmjn.sty, and ot1mjnx.fd which have been created using my virtual font installation packages mathinst and vfinst, which you can find on your local CTAN archive. (In these files, I assume that the Janson fonts have been renamed and installed according to the Berry font naming scheme, in which mjn designates Monotype Janson. These styles also assume the presence of the MathTime fonts, but unless you attempt to typeset mathematics, this should generate angry warnings but nothing else. If you plan to typeset with these fonts using \TeX or plain Omega, you will also need Damian Cudgley's plain font selection scheme, in the file pdcfsl.tex, which is part of the pdcmacs package on CTAN.) The file genesis.tex, which generated figure 11, also uses proprietary fonts, but even if you possess these Hebrew Hadassah fonts, you would need special \TeX macro files to properly typeset this figure.

38
Caveat!

The program _odvips_ suffers from at least one minor bug, which will bite you if you attempt to recreate any of the .eps files that are part of this manual. Experienced users know that you can create .eps from a one-page document with a command of the form

```
odvips -E -o foo.eps foo
```

where _foo.dvi_ has previously been created by Omega. For proper inclusion, bounding box dimensions have to be present, and these four numbers are calculated _wrongly_ by _odvips_. Before re-creating _foo.eps_, say, make sure you open the old file with a text editor, record these numbers, and then edit them into the new such file. The _BoundingBox_ comment which contains this information appears within the first half-dozen lines in any .eps file.
Figure 21: A portion of Ecclesiastes.