

# Printing Transliterated Indian Language Documents

**itrans**

version 5.30

©1991–2001 Avinash Chopde

Home page: <http://www.aczoom.com/itrans/>

---

This software is a pre-processor, it converts input representing Indic Text to the actual Indian Language Script, for printout or for display on Web pages. The input is uses English letters, so it can be created using any simple text editor.

This document describes the use of the program *itrans*, which may be used to generate Devanagari, Tamil, Bengali, Telugu, Gujarati, Kannada, Punjabi (Gurmukhi), or Romanized Sanskrit output from input in an english transliterated form.

Many examples of ITRANS use are available on the web, archives for different purposes are available. One such project is a collection of Hindi Film Song Lyrics; another project is a collection of Sanskrit documents. There are also tools on the Web that make it much easier to use ITRANS - you can use ITRANS over e-mail, or use your WWW Browser to type ITRANS text, etc. Links to many such sites are available through the ITRANS home page at and you can also use WWW search tools (use the keywords ITRANS and *indian*) to search the Web.

## 1 *itrans* Mechanism

*itrans* works by assigning every Indian language letter an english equivalent. Transliteration tables mapping devanagari or tamil or telugu, etc. characters into english equivalents are provided in the reference documents for each language, all of which are described in following sections of this document.

*itrans* scans the input text for consonants, vowels, and special forms. Consonants suffixed with vowel codes create a complete composite character. A consonant may be suffixed with one or more consonants to create ligature forms. All this transliteration is automatically handled, and appropriate Indian language characters are produced. If ligatures exist for a particular combination of consonants, they will be used automatically. If a ligature does not exist for some combination of the consonants, half-forms of the consonants will be used. The user can also override the ligature mechanism so that even if a ligature exists, the half forms of consonants will be used. Some languages such as tamil do not have any ligatures, in that case the appropriate action is taken — for tamil, a dot is printed on top of a consonant if it is not followed by a vowel form.

All these features make *itrans* an highly customizable and easy-to-use package. Even the transliteration map given here is not mandatory—the user can always edit the lexical source file and provide whatever mapping desired.

*itrans* is just a transliteration/char composition package. The task of actually placing characters on the page and spacing them correctly is left to other programs, such as  $\text{\TeX}$ .  $\text{\TeX}$  is the preferred interface, but, other modes are available for direct Text output (suitable for HTML output or Unicode UTF-8 output). ITRANS version 5.1 added a new output mode - direct text HTML output - which allows ITRANS output in certain scripts to be directly viewed on WWW browsers

---

<sup>0</sup>Last modification: December 4, 2009

such as Netscape (version 3.0 or newer). ITRANS version 5.3 added support for Unicode (UTF-8) output. These additional modes are available only for certain languages - not all are supported. Consult the language specific documentation for supported modes for each font.

## 2 Input Format

*itrans* makes use of an IFM file — Indian language font metric file, which is a ASCII file containing descriptions on how to generate the Indian language characters from the basic characters available in the font.

*Sidenote:* This IFM file format is an *itrans* specific concept, it allows all character composition directives to be loaded in at runtime, making it easy to support many different Indian languages. The IFM file is an ASCII text file, and all IFM files end in the suffix `.ifm`.

*itrans* scans through the input text, and copies everything to the output unchanged, except for portions between marker words, such as `#marathi` and `#endmarathi`. Some eight–ten different marker words are available, see the section 2.4 for more information. All english text between these words is mapped into Indian language characters, based on the transliteration map in the IFM file.

At the beginning of the input file, the user has to specify the IFM file, and the name of the  $\TeX$  or PostScript or HTML command that changes the font to the Indian language font. For example, if the IFM file is named `dvnc.ifm`, and the font is available through the `\devnf`  $\TeX$  command, the following two lines should be present in the input file:

```
#marathiifm=dvnc
#marathifont="devnf
```

This also assumes the user will be using the markers `#marathi` and `#endmarathi`, see 2.4 for all the other language markers and commands.

Once the above initialization is made, the `#marathi` marker then specifies the beginning of the marathi transliterated text, and makes use of the specified IFM file (`dvnc.ifm`). At that point, *itrans* also outputs the command (`\devnf`) specified in the `#marathifont` directive. This command should change the font to the devanagari font, and may do other things such as change the baselineskip length, etc.

Note that both the  $\TeX$  interface and the Direct Text interface (both PostScript and direct Text HTML mode) follow identical input text requirements. For further examples, see the sample documents provided. All transliterated files have been given the file extension `.itx`. [Older ITRANS version also used `.ips` extensions for direct PostScript output, but since ITRANS version 5.0, the `\char35` output command has been added which allows specifying the output mode in the input file itself.)

### 2.1 The “`#include=`” command

Itrans accepts an “include” filename command in the input. Syntax:

```
#include=<filename>
```

This command can appear anywhere in the input document, and Itrans behaves as if the contents of that file were actually present at that point in the document.

This “include” command can be nested in multiple files (to a compiled-in maximum).

### 2.2 The “`#output=`” command

This command should be the first ITRANS command in any input file, that is, it should appear before any other `#<command>`. It can be used to direct ITRANS to produce kind of output -

TeX, or PostScript, or direct Text HTML output. So, instead of using arguments to ITRANS (such as -P or -7 or -8 or -U), users can include this command in the input file itself, making it clear what the input file is to be used for. The four valid options for this command are:

```
#output=HTML_7
#output=HTML_8
#output=UTF_8
#output=PostScript
#output=TeX
```

### 2.3 The “#endfont=” command

This command is generally of use in HTML output modes only, though if needed, it could be used in any mode.

This command allows the user to specify a string that will be echoed to the output file whenever any #end<language> is seen.

For example, it is useful to use this in HTML documents, where every end of Indic script needs to print out a </FONT> command, and this can be done automatically by specifying:

```
#endfont=</FONT>
```

This, in conjunction with something like:

```
#hindifont=<FONT FACE=name SIZE=size>
```

makes it easy to use ITRANS in HTML output mode.

Note that there is single #endfont command, and it applies to all language markers.

### 2.4 #<lang>, #end<lang>, and ##markers

You can use any of these marker sets to delimit the Indian language text. The marker names indian/marathi/hindi/tamil do not actually do anything by themselves, but make use of the corresponding command names to load in the IFM file or output the font changing command string. So, use any one of the sets you feel suits your needs best, each can be used for every language supported for ITRANS, the marker is just used to enter ITRANS mode, the actual language is always correctly recognized based on the IFM file. Since the marker words are all long, a shorter version of markers is also available, and the short markers are enabled by default, though it is possible to turn them off.

#### 1. indian marker.

To set the IFM file name: #indianifm=XXX.ifm

To set the font command name: #indianfont=YYY

Start Marker: #indian

End Marker: #endindian

#### 2. hindi marker.

To set the IFM file name: #hindiifm=XXX.ifm

To set the font command name: #hindifont=YYY

Start Marker: #hindi

End Marker: #endhindi

3. `sanskrit, marathi, tamil, telugu, bengali, gujarati, roman, kannada, gurmukhi`.

Just as for hindi and indian, there are markers for all these languages. Follow above examples, replace indian with `<language>` as required.

4. `##` short marker.

These markers are activated by default. To turn them off, use the `#ignoreshortmarkers` command.

#### 2.4.1 “`#useshortmakers`” and “`#ignoreshortmarkers`”

The short marker is a toggle marker. When scanning text in non-ITRANS mode (non Indic text), if a short marker is seen, it implies restoring back ITRANS processing, using whatever language marker was last encountered. Then, the next seen short marker implies ITRANS should exit processing of the indian language text.

This, if the input text has the following input:

```
#hindi          <some text>          #endhindi
<some more text>
##             <short marker text>      ##
```

then the first short marker seen above will be considered equivalent to `#hindi`, since that was the last ITRANS marker seen in the text at that point.

If the short marker is the first marker seen in the text, i.e., there was no other ITRANS marker seen until that point, then the `##` shortmarker will be taken to be equivalent to `#indian`.

#### 2.5 “`#usecsx`”

The CS/CSX input encoding can be accepted along with the ITRANS encoding when this command is used. See the `icsx.itx` document for more information on CS/CSX support in ITRANS.

#### 2.6 “`#endwordvowel=`”

See the section 9.5 for info on this command.

### 3 **T<sub>E</sub>X** Interface

*itrans* can accept T<sub>E</sub>X input and generate T<sub>E</sub>X output for all languages for which either a PostScript font description or a Metafont description is available.

For example, the **ॐ** character in the table in the document *dvnc.itx* was produced by this input text:

```
{#marathi aa #endmarathi}
```

Thus, the character *aa*, when bracketed between `#marathi` and `#endmarathi` produces **ॐ**, when the appropriate IFM file name and font command name have been set correctly (as mentioned in the previous section).

For further examples, see the sample documents provided. All T<sub>E</sub>X transliterated files have been given the file extension *.itx*.

Normally, all english text between the words `#marathi` and `#endmarathi` is mapped into marathi characters, except for any word following a backslash character. Thus, you can include

T<sub>E</sub>X commands in the transliterated text portion, the only restriction is that the command should be made up of letters and numerals only.

Examples: `\indent` or `\hskip1in` or `\kern0.4em`, etc. Note that in the sample documents shown, T<sub>E</sub>X commands are usually surrounded by curly braces—they used as scope delimiters only, are not absolutely necessary, but are recommended.

### 3.1 T<sub>E</sub>X Requirements

To use this pre-processor, you must have the following tools:

1. T<sub>E</sub>X tools, including the capability to output dvi files using PostScript fonts (if you need to use the devnac devanagari font). This implies having *dvips*, release 5.41 or newer, a program written by Tomas Rokicki. If you don't have it, see the Frequently-Asked-Questions in the newsgroup comp.text.tex for details on how to get hold of *dvips*. Actually, if you only need to use tamil or Frans Velthuis's Devnag font, or telugu, you don't need a dvi-to-ps converter. Those fonts are described in Metafont, thus any dvi-to-your-printer converter will be able to handle it (once you run Metafont and generate the printer specific files).

For the IBM PC, the emTeX package includes the *dvips* executable. *dvips* for the PC also resides in the SIMTEL archives.

2. ANSI C compatible compiler, and optionally, the flex and yacc tools. This program is provided with the source code, and a makefile. The makefile is specific to my machine, you may have to edit it to get it working correctly on your machine. If you do not have an ANSI C compatible compiler, you will have to make the "noansi" object, see the makefile for details; it automatically de-ansifies all the C sources. Also, if you need to modify the `ilex.l` or `iyacc.y` source files for your specific needs, you'll need the flex and yacc tools on your system.

The ITRANS package also contains an executable for MS-DOS machines, thus for MS-DOS machines, you need not go through the compilation step.

### 3.2 L<sup>A</sup>T<sub>E</sub>X Style Files

In ITRANS release 5.2, a new style file was added *itrans.sty*. This is the file created by Ross Moore to allow ITRANS documents to be processed through latex2html, but this style file is useable for normal L<sup>A</sup>T<sub>E</sub>X documents also.

This style file should be loaded with the `\usepackage` command, and there are many options available with this package.

**preprocess** This is applicable only when using latex2html, and it has no effect when running normal L<sup>A</sup>T<sub>E</sub>X. So, it is probably ok to always use this option even if you never plan on running latex2html.

**devanagari** This will load the `idevn.tex` file, which has some macros that are useful for using Devanagari with T<sub>E</sub>X. `idevn.tex` will actually be loaded even if no option is provided, so the devanagari option is not really necessary at this time (ITRANS release 5.2).

**telugu** This will load the `itrnstlg.tex` file, which has some macros that are useful for generating some special Telugu characters.

**other options** *itrans.sty* has many other options, take a look at that file to view all of them, but in practice, you may not need to use any of those options yet.

Example lines to load itrans.sty:  
`\usepackage[preprocess]{itrans}` or  
`\usepackage[preprocess,telugu]{itrans}`

## 4 Direct PostScript Output

*Note: This section is applicable to the PostScript fonts only: devanagari font devnac, Gujarati ItxGuj, Bengali ItxBeng, Devanagari Xdvng, and Romanized Sanskrit NCS\_CSX. The tamil font wntml, the devanagari font devnag, the telugu font tel and the kannada font kan cannot be used for this purpose since they are not PostScript fonts.*

As mentioned earlier, the Direct PostScript Output interface is an extremely primitive interface for printing, but will suffice for printing documents containing only Indic Script text (no english), and requiring no typesetting features such as centering, right flush text, etc. In addition to generating the devanagari characters, this method preserves the line breaks and spaces in the input text. So, unlike in the  $\TeX$  version, which programatically decides where to break a line, here you have to include a end-of-line in the exact spot where you desire a new line to start in the output. And, if you need indented lines, you have to add spaces to simulate horizontal skips.

The previous section titled *Input Format* applies to this interface, too. Thus, the user needs to specify the IFM file name, the font command name, etc before writing text between the Indian language markers such as `#marathi` and `#endmarathi`. For further examples, see the sample document provided in the ITRANS archive, file may be named something like `s1.ips`. The PostScript prologue for itrans is in the file *itrans.pro*. Check the file out, it contains some useful PostScript procedures.

### 4.1 Direct PostScript Output Requirements

1. All that is required is the capability to print PostScript files. Note that the *devnac* font used is a user defined PostScript font (Type III Font), so the printer (or the RIP used) must be capable of rendering Type III fonts if you need to use this font.

## 5 Direct Text Output – for HTML

This is a new mode added in release 5.1, which just like the PostScript mode, does not offer any typesetting functionality. In fact, this mode is provided for generating text that can be displayed on any WWW browser such as Netscape (version 3.0 or later). Not all languages are supported in this mode — it requires availability of a TrueType font, and right now (ITRANS 5.10) the following languages are supported: Romanized Sanskrit, Gujarati, Bengali, and Devanagari scripts.

Two types of HTML output are supported: 7-bit ASCII (which uses the `&#nnn` HTML codes for character codes 127 to 255), and 8-bit text output (which outputs character codes 127 to 255 directly). I think 7-bit output should be more portable, but in practice, I found it to create problems on Macintosh computers, so, I use 8-bit output in the examples.

Three platforms have been tested: Unix/X11R6 (Type 1 fonts - `*.pfa` or `*.pfb`), Macintosh (TrueType - `*.hqx` files), and Windows (TrueType - `*.ttf`). All the fonts required are present in the ITRANS/lib/fonts directory of the ITRANS package, the user has to install the appropriate font on his or her system to allow viewing of the ITRANS output in the native Indic Script.

Sanskrit/Hindi/Marathi	<i>udvng.ifm</i>
Gujarati	<i>uguj.ifm</i>
Bengali	<i>ubeng.ifm</i>
Tamil	<i>utml.ifm</i>
Telugu	<i>utel.ifm</i>
Kannada	<i>ukan.ifm</i>
Romanized Sanskrit	<i>uroman.ifm</i>
Gurmukhi	<i>ugur.ifm</i>
Malayalam	<i>umal.ifm</i>
Oriya	<i>uoriya.ifm</i>

Table 1: **ITRANS: Unicode IFM files.**

## 5.1 Direct Text (HTML) Output Requirements

The font should be available in Type 1 or TrueType format, and must be usable in regular word-processors (without ITRANS).

## 5.2 HTML Output – Unicode – UTF-8

ITRANS 5.3 added support for Unicode output using UTF-8 encoding for the output.

The UTF-8 text output can be displayed on any Web browser or loaded into any text editor that can handle UTF-8 text. To view the text in Indian language scripts requires availability of an Unicode font, for the specific language. ITRANS does not come with any Unicode fonts, it is expected that in due time, there will be freeware Unicode fonts for all Indian languages available on the Internet.

The Oriya and Malayalam scripts are supported through the Unicode interface only - these scripts are not supported in the T<sub>E</sub>X or PostScript interfaces.

## 6 Program Options

See the manual page for details regarding the program options need to be specified to itrans to make it run in the Direct PostScript Output mode (option -P), or direct Text (HTML) mode (options -7 or -8 or -U). Without any of these options, the T<sub>E</sub>X interface is assumed. Running itrans also requires that the environment variable ITRANSPATH be set correctly: it should contain the list of directories contain all the \*.ifm, \*.tfm, and \*.afm files that itrans may need. Again, the manual page has more details on this. If at any time itrans prints out a error about some file not found, it usually implies that ITRANSPATH has to be modified to include the directory of that file.

## 7 Known Problems

1. Version 4.00+ *itrans* can handle simple comments in the transliterated text portion of the input file. % character begins a comment, the end-of-line terminates a comment. Handling of comments is not fully coded yet, and has some problems in that the % always starts a comment, only exception being when % is preceded by a backslash (\). Of course, there are

many other instances where % should not begin a comment, those are ignored for now. (And may be ignored for ever, recognizing comments correctly would require too much effort.)

2. In the T<sub>E</sub>X interface, characters with any non-zero Y offset are not correctly printed in some cases, when the PostScript font Devnac is used. One such case is the da-ra ligature, words such as *draaviiDa* (द्रावीड). (Another case is the ha-u form हु, see how it is handled in the sample input file *nehru.itx*.) Note that it is only in certain cases that the word is printed incorrectly, in most cases it is handled correctly. Usually, when the word appears near the end of the line, T<sub>E</sub>X (or dvips, but that sounds improbable) inserts a kern (a glue factor ?) just before the character with a non-zero Y offset, and the word appears squashed up or pulled apart at that point. Have no remedy for this, only workaround is to force a line break before the problem word, it usually sets everything right.

I have never encountered this problem when using Frans Velthuis's Devnag font with the *itrans* package, therefore this problem is probably related to the use of PostScript fonts in T<sub>E</sub>X. If you do notice this problem with the Devnag font too, please let me know.

## 8 Output Languages Supported

### 8.1 Devanagari Output

Two devanagari fonts are supported. The bundled in devanagari font is a PostScript font, called *devnac*. *devnac* was developed (and is under further development) by Avinash Chopde. The other font is a Metafont font called *devnag* and has been developed by Frans Velthuis. More details regarding *devnag* can be found in the file *dvng.itx*, more details regarding *devnac* can be found in *dvnc.itx*. Transliteration tables for each language are available in those files, too.

*devnac* is a PostScript Type III font. This font can be used with both the T<sub>E</sub>X interface, and the direct PostScript interface mode of *itrans*.

The font is named *dnh* in the T<sub>E</sub>X interface, and variations are also available, named *dnho*, *dnhrc*, *dnhre*. In the direct PostScript interface, the generic font changing commands *normalfont*, *slantfont*, *compressedfont*, *expandedfont*, etc have to be used. See the file *itrans.pro* to get a handle on the workings of the above PostScript commands.

This devanagari font tries to be a all-encompassing font, for the hindi, marathi, and sanskrit languages.

See the reference document *dvnc.itx* for the transliteration map and example texts. The IFM file for this font is *dvnc.ifm*.

*devnag* is a Metafont font, and thus can only be used with the T<sub>E</sub>X interface of *itrans*. This font is supported by *itrans* in a limited manner, in that not all ligatures are available for use. Many ligatures are archaic, so I've left them out. Again, the document *dvng.itx* contains complete details and lists of the ligatures that are used and that are ignored by *itrans*. The IFM file for this font is *dvng.ifm*.

A derivative of the *devnag* font, called *xdvng* is also available, for use with the direct text HTML output mode of ITRANS. See *dvng.itx* for more info on this font.

### 8.2 Tamil Output

The tamil font is a Metafont font, and was created at the Humanities and Arts Computing Center of the University of Washington, USA. The Metafont files for this font are also bundled with the *itrans* package.

This font can only be used with the T<sub>E</sub>X interface.

The Metafont descriptions of the font are provided, you can use them to generate the PK files of any font size desired. The Metafont programs for three sizes: 10, 12, and 17 point sizes are provided (*wntml10.mf*, *wntml12.mf*, and *wntml17.mf*, respectively).

This font was developed at University of Washington, and I would like to thank them for making the font available as freeware.

See the reference document *tamil.itx* for the transliteration map and example texts. The IFM file for this font is *wntml.ifm*.

### 8.3 Telugu Output

The single font *tel* is currently supported. It is a Metafont font. This font can only be used with the T<sub>E</sub>X interface.

The telugu font is from the TeluguTeX package (which is ©Lakshmi V.S. Mukkavilli, 1991). This is a Metafont font.

See the reference document *tlgutx.itx* for the transliteration map and example text. The IFM file for this font is *tlgutx.ifm*.

### 8.4 Bengali Output

ITRANS 5.1 added support for the TrueType (and PostScript Type 1) Bengali font named *ItxBeng*, and in ITRANS 5.2, Jaijeet Roychowdhury added support for the Bengali font from the BWTI package

The IFM file for the *ItxBeng* font is *itxbeng.ifm*, and the IFM file for the BWTI font is *bnbeng.ifm*.

The *ItxBeng* font has been provided by Shrikrishna Patil, while the BWTI Metafont Bengali font has been developed by Abhijit Das.

[ITRANS 4.x supported a LaserJet Softfont called *SonarGaon*, but that has now been removed from ITRANS.]

See the file *beng.itx* for documentation on both the bengali fonts supported by ITRANS.

### 8.5 Gujarati Output

The gujarati font *ItxGuj* is a PostScript Type 1 font. and this *ItxGuj* font has been provided by Shrikrishna Patil. *itxguj.pfa*, *itxguj.afm* are the Type 1 PostScript files, and *itxguj.tfm*, *itxgujo.tfm*, *itxgujrc.tfm*, *itxgujre.tfm* are the TFM files for use with TeX. The user manual for this font is in the file *gujdoc.itx*.

The IFM file for this font is *itxguj.ifm*.

### 8.6 Kannada Output

The single font *kan* is currently supported. It is a Metafont font. This font can only be used with the T<sub>E</sub>X interface.

The Kannada font is from the KannadaTeX package (which has been developed by G. S. Jagadeesh and Venkatesh P. Gopinath). This is a Metafont font.

The IFM file and other support in ITRANS for Kannada was added by Raghunath K Rao.

See the reference document *kantex.itx* for the transliteration map and example text. The IFM file for this font is *kantex.ifm*.

## 8.7 Punjabi Output

Anshuman Pandey added support in ITRANS for the Gurmukhī Postscript font ‘Punjabi’ (`pun`) (designed by and copyright Hardip Singh Pannu).

`pun.pfa`, `pun.afm` are the Type 1 PostScript files, and `pun.tfm` is the TFM file for use with TeX. The user manual for this font is in the file *pundoc.itx*.

The IFM file for this font is *pun.ifm*.

## 8.8 Romanized Sanskrit Output

The Romanized Sanskrit font NCS\_CSX is a PostScript Type 1 font, details about this are present in the *romanctx.itx* file.

This font can be used with all three output modes of itrans – TeX, PostScript, or Direct Text Output. The Direct Text Output requires using the TrueType version of this font.

The PostScript Type 1 font files are in `ncpr____.afm`, `ncpr____.pfb`, `ncpi____.afm`, `ncpi____.pfb`. TeX TFM versions are available in `ncprcsxp.tfm`, `ncpicsxp.tfm`. TrueType versions are available in `ncpr____.ttf`, `ncpi____.ttf`.

# 9 Usage Hints

## 9.1 Ligature suppression

As mentioned earlier, the system automatically uses ligatures whenever possible. For example, since the `ta-ta` ligature exists, input text of the form `tti` is printed as `त्ति`. If instead you need it to be printed as `त्ति` you have two choices. One, if you never want the `ta-ta` ligature to be used, you can edit the IFM file and comment out all the lines that refer to the `ta-ta` ligature. (The IFM file is a text file, for more information, see the technical documentation in *tech.tex*.)

On the other hand, if you do want to keep the ligature, except in a few locations in the input text (say for small point type), or if you do not want to edit the IFM file, you can use the ligature inhibitors `{}` to prevent a ligature from being used. Whenever the `{}` characters are inserted between two consonants, *itrans* refrains from using the ligature (if it exists, if it does not, then these characters have basically no effect). Instead, the half-forms of the consonants (as appropriate) are used. Thus, even if the IFM file contains the `ta-ta` ligature, the input text `t{}ti` always appears as `त्ति`.

## 9.2 Breaking lexical scan

Use the character `_` after a consonant letter to break the lexical scan. This becomes necessary because of the default behavior of the scanner, which tries to match the largest possible input pattern. Thus, when you write `ai` in the transliterated text, it comes out as `ऐ` in marathi or `ஐ` in tamil. Now, that is just what you want for the `ai` vowel, but what if you wanted it to be scanned as two vowels: `a`, followed by `i`? In such cases, you need to break the lexical scan, by following the first vowel, the `a` character, with the `_` character to stop the scanner from associating `i` with the `a` preceding it. So, `a_i` in the input text results in: `अइ` for marathi and `அஇ` for tamil.

This is a thing to watch out for in all cases where some character has a multiple letter mapping, and each letter by itself also represents some other character. In the above example, `ai` is the two letter map, and both `a` and `i` represent other vowels.

Note that if you wish to suppress any ligature, you should use the {} letters consecutively, as explained in the previous paragraph. Using \_ to follow some consonant allows a ligature, if it exists for the consonant pair, to be used. (Of course, tamil does not have any ligatures.)

To get a printable underscore in the Indian language text, follow normal T<sub>E</sub>X usage—use backslash underscore—\\_.

Since ITRANS supports many Indian languages, it has a large list of input tokens that are mapped to consonants. For example even if ai is not a vowel in one of the languages supported by ITRANS, it still is always recognized as the ai vowel, and thus if you need it to be recognized as the a and i vowels, use the \_ character as mentioned above.

### 9.2.1 List of all ITRANS recognized English letters

Here's the list of ASCII input characters that are special to ITRANS:

```

0 1 2 3 4 5 6 7 8 9
.a .c .N .n ^r .D .Dh A AUM
a aa ai au b bh ch chh D Dh d dh dny E e f G g gh GY
H h I i ii J j jh JN K k kh kSh L L^i L^I LLi LLI l ld
M m N "n ~n n ^n N^ ny O o OM p ph q
.r ^r .R R R^i R^I RRI RRI r S SRI s sh shh
T Th t th U u uu v x Y y z
#<language>
#end<language>
#<language>ifm
#<language>font
#output=HTML_7
#output=HTML_8
#output=UTF_8
#output=TeX
#output=PostScript
#ignoreshortmarkers
#usesshortmarkers
#usecsx
#ignorecsx
#include=
#endwordvowel=a
#endwordvowel=.h
##      {}      -      '

```

Version 4.0 and onwards added support for the Classical Sanskrit and Classical Sanskrit Extended encoding, which adds one ASCII character to the above list (c) and many non-ASCII characters, see the document `icsx.itx` for more details.

### 9.3 Punctuation Issues

The fonts provided may be missing some or all punctuation characters, some may also be missing numbers. For example, the tamil font used, `wntml`, does not have any numerals or any punctuation characters. The devanagari font does have numbers and some punctuation marks (the double-quote is missing, for example).

For devanagari, when using  $\TeX$ , use the `idevn.tex` (automatically loaded by `itrans.sty`, use the `\usepackage{itrans.sty}` command in  $\LaTeX$ ) which provides modes for setting a devanagari font, and offers many commands that ease devanagari input. See the ITRANS Song Book documents for examples of `idevn.tex` usage, or, check out the sample document file `sample.itx`.

In all punctuation problems, you can always get the required punctuation or digit character by ending the Indian language transliteration scope (using one of the endmarkers), then printing the required punctuation mark, and the restarting the transliteration by using the start marker.

An easier method, in  $\TeX$  is to make use of the math mode for numbers. It is usually sufficient to use the  $\$$  enclosing scope to make numbers print correctly, since a  $\$$  enters mathmode and uses the math fonts. For punctuation marks, the user needs to explicitly change fonts: example:

```
#marathi.....{\rm ;}.....#endmarathi.
```

But that is preferable over this form:

```
#marathi.....#endmarathi; #marathi.....#endmarathi.
```

## 9.4 Multi-consonant conjuncts

How does `itrans` handle ligatures with more than two consonants ? For example, `shhTmii` contains three consonants. This ligature produces **ह्मी**, the way `itrans` works is as follows: Beginning with the first consonant in the list, `itrans` checks if a double-consonant ligature has been defined for that consonant and the next one in the list. If such a character exists, then it is used and both consonants are consumed, and `itrans` repeats the procedure for the next consonant.

There is one exception to the above rule: if at all possible, the last two consonants are handled together, that is if a ligature of the last two consonants exists, that is used over the pairing that would result from the above method. Example: `shhTrii` produces **ह्री**, both `shha-Ta` and `Ta-ra` ligatures exist, but since the consonants `Ta` and `ra` are the last two consonants, that ligature is used over `shha-Ta`.

Of course, this default behavior can be changed by appropriately placing the ligature inhibitor sequence, `{}`. Example: `shhTr{}ii` produces **हरी**.

Also, in ITRANS version 5.10, ITRANS now supports direct specification of 3 or more consonant conjuncts in the `.IFM` file. So, if the above given description leads to incorrect output, we can now fix it by providing explicit rules for 3 or more multi-consonant conjuncts in the `IFM` file.

## 9.5 Word endings and the `#endwordvowel` command

To increase readability of the Indian text (in english), each word for the devanagari and bengali language is assumed to end in a vowel. Each Indian language letter can be generally written as  $C + [C + [C + [...] + C]] + V$ , where  $C$  is a consonant, and  $V$  is a vowel. When a consonant appears at the end of the word, and the vowel is “a”, then it is not necessary to include the final “a”—if a word ends in a consonant, ITRANS will automatically add the vowel “a” to the final consonant—for devanagari and bengali only. Thus, in hindi, instead of writing “hama”, you can write “ham”, which is how the word **हम** is pronounced anyway. To add a halant to any consonant, use “.h”, thus “ham.h” produces **हम्**.

Use this default mechanism only when it increases the readability, for example in hindi it is better to write “ek”, “is”, “tab”, instead of “eka”, “isa”, “taba” for **एक**, **इस**, **तब**. But, it is better to include the “a” in words such as “manushhya”—**मनुष्य**.

This default mechanism is activated only for the devanagari and bengali input. (The keyword `DEFAULTVOWEL` in the `IFM` file directs whether the last consonant in a word should be considered as paired the half-form, or the a-form, if the form is left unspecified.)

The above default working can be avoided by specifying the vowel to end words in the input document.

```
#endwordvowel=a
```

```
#endwordvowel=.h
```

The first command makes the default vowel be “a”, which the second command will make the default vowel a “half-form”—viraam. The endwordvowel command will override the DEFAULT-VOWEL specification in the IFM file. So, for sanskrit, it is useful to include this command in the input file:

```
#endwordvowel=.h
```

which puts a viraam at the end of any word that does not end in a vowel.

---

## 10 Author

ITRANS has been developed by Avinash Chopde.

E-mail: [avinash@aczoom.com](mailto:avinash@aczoom.com).

Home page: <http://www.aczoom.com/>

Over the years, numerous people from all over the world have made important additions to the ITRANS package as it exists today. It is not possible to thank everyone here, but I have tried to include correct acknowledgements in appropriate documents - the CHANGES text file lists all changes in each release and the names of persons who contributed each extension, and each of the language document files also acknowledge the help received from particular individuals.

I would like to thank all the users of ITRANS, past and present, who send me additions and extensions and send in bug reports. Thanks to all!