1 Additional Features

Q: There are now already two LaTeX implementations of the Tibetan language. Why is there now a third one?

A: Like other typesetting programs, the above implementations make use of predefined ligatures only. If you try to enter prayers containing Sanskrit syllables into such a system, you will get into severe trouble. Therefore, I added now a completely new transcript preprocessor having the following extra features:

Compatibility: As far as I know, it supports all control sequences of the above implementations, plus the ACIP transcript. If you have yet another transcript, you can create your own transcript definition file, hopefully without having to modify the C code.

Inline LaTeX commands: All commands that do not contain spaces or such can be typed without using the mode switching characters(`%%`). Even the tabbing character (`&`) is transparent. So you can easily typeset tabular environments and line breaks.

Generic Ligatures: If this preprocessor encounters any ligatures which are not predefined, it tries to compose them automatically. There is still much work to be done on this:

- There are some spacing problems to solve on LaTeX level
- Adding intelligence for character mutations
- Font definition for character fragments
- Font definition for vertically smaller characters

Especially syllables like །ཏཏྟཱུ་ར། do not look very nice and need too much vertical space. But in many cases, the output is quite readable and the input very simple (e. g. `/tutt'a ra/`).

2 How to use it

2.1 General Command Line Handling

In fact, it is very similar to the previous implementations: You have the usual LaTeX procedure plus an additional preprocessor program between editing and LaTeXing, which interprets a transcript described below, generating Tibetan characters and ligatures.

There is some extra complication concerning the command line parameters:

```
STARTING THIS PROGRAM
---------------------
In order to translate ACIP to LaTeX, use

ttt acip.ttd latex.ttd input.tib output.tex

where acip.ttd   is the input  transcript definition file
.     latex.ttd  is the output transcript definition file
.     input.tib  is your ACIP input text
.     output.tex is the output file to be processed by latex

On Linux, you will enjoy the feature of magic line support:
In your input.tib file, add the following two lines at the
VERY TOP (replacing the respective filenames):

#!/bin/sh
ttt acip.ttd latex.ttd $0 output.tex; latex output; exit

Due to the magic line at the top, (ba)sh will think that your text
input.tib is a shell script, executing the 2nd line.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
This program will ignore the FIRST AND THE SECOND
line if the FIRST line starts with #!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Do not put anything, even not comments above these lines.
$0 is automatically replaced by the file name (e.g. input.tib).

Then turn your input.tib file into an executable by
chmod 755 input.tib

Now, you can start this program by entering
input.tib
at the shell command prompt.

DOS users, please write an apropriate *.BAT file (without $0).
Windows users may set the command line options as program properties.
```

2.2 Which ttd files to use

At present, `input.ttd` and `acip.ttd` can be used as input transcript definitions only. Use `input.ttd` for any texts supplied by Jeff Sparkes and Sam Sirlin and this package (ending in `.tib`). `acip.ttd` only for ACIP texts. You easily recognize them by their having an UPPERCASE TRANSCRIPT.

`tex.ttd` and `latex.ttd` are the output transcript definitions. Use `latex.ttd` **only** for ACIP texts. It includes a document preamble and postamble for LaTeX. In all other cases, I recommend you to include the preamble and postamble into the source file and use `tex.ttd`.

3 Transcription

3.1 Note

The upper lines show the transcript used by Jeff Sparkes and Sam Sirlin. The lower lines show the ACIP transcription. Please note the difference marked by (`!`). Of course, punctuation and spacing are completely different. For LaTeX, I recommend the transcript in the upper lines, since this transcript definition supports the full power of the ligature generator.

3.2 Single Characters and Vovels

ཀ་	ཁ་	ག་	ང་	
ka	kha	ga	nga	
KA	KHA	GA	NGA	
ཅ་	ཆ་	ཇ་	ཉ་	
ca	cha	ja	nya	
CA	CHA	JA	NYA	
ཏ་	ཐ་	ད་	ན་	
ta	tha	da	na	
TA	THA	DA	NA	
པ་	ཕ་	བ་	མ་	
pa	pha	ba	ma	
PA	PHA	BA	MA	
ཙ་	ཚ་	ཛ་	ཝ་	
tsa	tsha	dza	wa	(!)
TZA	TSA	DZA	WA	(!)
ཞ་	ཟ་	འ་	ཡ་	
zha	za	'a	ya	
ZHA	ZA	'A	YA	
ར་	ལ་	ཤ་	ས་	
ra	la	sha	sa	
RA	LA	SHA	SA	
	ཧ་	ཨ་		
	ha	a		
	HA	A		

ཨ་	ཨི་	ཨུ་	ཨེ་	ཨོ་	ཨཾ་	ཨཱ་
a	i	u	e	o	*	aa
A	I	U	E	O		

ཊ་	ཋ་	ཌ་	ཎ་	ཥ་	ཀྵ་
txa	thxa	dxa	nxa	shxa	kshxa

3.3 Punctuation Marks and Spacing Control

།	༈	་	suppress"་"	shad space	་	ཿ
/	!	,	#	|	\swasti	:
,					* (?)	

4 Examples

4.1 Explicit Ligature Control

གྱིན་	གཡིན་	གྱས་	གཡས་	དདྷ་	དྡྷ་
gyin	g-yin	gyas	g-yas	ddha	-ddha

4.2 Generic Ligatures

ཨོཾ་	མ་	ཎི་	པདྨེ་	ཧཱུྃ་	ཧྲཱིཿ	ཨོཾ་མ་ཎི་པདྨེ་ཧཱུྃ་ཧྲཱིཿ
o*	ma	nxi	padme	h'u*	hr'i:|	Yet another spacing bug!

ན་	མོ་	གུ་	རུ་	བྷྱཿ	ན་མོ་གུ་རུ་བྷྱཿ
na	mo	gu	ru	bhya:|	

ན་	མོ་	བུདྡྷ་	ཡ།	ན་མོ་བུདྡྷ་ཡ།
na	mo	buddha	ya/	

ན་	མོ་	དྷརྨཱ་	ཡ།	ན་མོ་དྷརྨཱ་ཡ།
na	mo	dharm'a	ya/	

ན་	མོ་	སངྒྷ་	ཡ།	ན་མོ་སངྒྷ་ཡ།
na	mo	sanggha	ya/	

།ཨོཾ་	ཨཱཿ	ཧཱུྃ།	།ཨོཾ་ཨཱཿཧཱུྃ།
/o*	aa:	h'u*/	