Wait Zar Hacker's Guide

version 1.8

Table of Contents

1. Introduction____ p. 1
  1.1 Who Is This Guide For?___________________________________ p. 1
2. How to Get Involved____ p. 1
3. What Sections Should I Read?_______________________________ p. 1
4. Hacking the Configuration File_____________________________ p. 1
  4.1 How WaitZar Searches for Config Files____________________ p. 2
  4.2 Introduction to JSON & Config File Syntax________________ p. 3
    4.2.1 JSON Syntax____ p. 3
    4.2.2 WaitZar's Use of JSON________________________________ p. 4
    4.2.3 Tracking Down Errors_________________________________ p. 5
  4.3 Settings____ p. 5
    4.3.1 The Settings Dialog__________________________________ p. 5
    4.3.2 Settings.Hotkey______________________________________ p. 5
    4.3.3 Settings.*____ p. 6
  4.4 Languages____ p. 9
  4.4.1 Languages.*____ p. 9
  4.5 Simple Examples: Common Configuration Tasks______________ p. 9
  4.6 Complex Example: Loading a Completely Different Language_ p. 9
5. The WaitZar Romanisation___________________________________ p. 10
  5.1 Default Romanisation____________________________________ p. 11
  5.2 Special Romanisation____________________________________ p. 15
  5.3 Shortcuts____ p. 16
  5.4 Dictionary Lookup_______________________________________ p. 17
  5.5 Known Font Issues_______________________________________ p. 17
6. Coder's Guide____ p. 18
  6.1 Getting the Source______________________________________ p. 18
  6.2 Compiling and Running Wait Zar from Source______________ p. 18
    6.2.1 Configuring the Eclipse CDT_________________________ p. 19
    6.2.2 Understanding the Different Configurations__________ p. 20
    6.2.3 Compiling____ p. 20
    6.2.4 Running____ p. 20
  6.3 Understanding the Code__________________________________ p. 21

6.3.1 Startup Events ____ p. 21
6.3.2 Bundled Resources ____ p. 21
6.3.3 Loading a Configuration ____ p. 21
6.3.4 Flow of Input ____ p. 21
6.3.5 The Help Keyboard ____ p. 21
6.3.6 The System Tray Icon and Menus ____ p. 21
6.3.7 Miscellaneous Interesting Classes ____ p. 22
6.3.8 Design Philosophy ____ p. 22
Appendix A: License ____ p. 22

1. Introduction

Welcome to Wait Zar, a syllable-level romanised input system for the Burmese language. To get the latest information on WaitZar, please go to the Google code page:

http://code.google.com/p/waitzar/

Wait Zar is a community project. It is open-source, and free. (This user's guide was typed using Wait Zar in Open Office.)

1.1 Who Is This Guide For?

This guide is for advanced users, WaitZar developers, and module developers. It explains how to configure and change WaitZar in great detail. If you only need to know how to use WaitZar for typing, you should read the **WaitZar User's Guide**, available online at:

http://waitzar.googlecode.com/svn/trunk/WaitZar%20User%27s%20Guide.pdf

Please be aware that this guide is nowhere near complete, and is occasionally inaccurate. In general, highlighted text indicates incomplete sections. Please contact us with any questions and we'll do our best to answer them.

2. How to Get Involved

(brief overview: hack the soruce code, hack the config files. For non-programmers: the romanisation, and bug testing/translating/documenting. Also make a distinction here between "core" WZ (the input method) and "general" WZ (the pluggable part). Note that Section 4 (romanisation) is only for "core".)

3. What Sections Should I Read?

(General information on each section. Also, list what sections each role (tester/documenter) should read.).

4. Hacking the Configuration File

Wait Zar can be customized to suit your needs. For example, many programmers use "Ctrl+Shift" to select rows of text. They may want to change the language hotkey to "Alt+Shift" —but anyone typing in Chinese will prefer to change it to something else ("Ctrl+Space" maybe). A great deal of configuration options will be available to Wait Zar users. This section details some of them.

In order to apply any of these custom configurations, you must unzip WaitZar.exe into a directory. You cannot just run it from within the zip file. You should also unzip config.txt and mywords.txt, although this is not strictly necessary.

4.1 How WaitZar Searches for Config Files

The first thing to know about configuring WaitZar is that there are several different configuration files. They reside in one of the following directories:

- **WAITZAR** — This is the directory where you extracted WaitZar.exe.
- **APP DATA** — This is the Windows-specified folder for application data. On Windows 7, this is found via:

  ```
  echo %USERPROFILE%\AppData\Local
  ```

- **DOCUMENTS** — This is the Windows-specified user documents folder. On Windows 7, this is found via:

  ```
  echo %USERPROFILE%\Documents
  ```

The various config files are listed in the following table, along with their priorities in case of duplicate settings. Note that some configuration settings allow users to change the locations of some of these files (particularly the word-list files).

File	***Priority***	***Contents***
DOCUMENTS\waitzar.config.json.txt	High	Initially empty. Users should add their own preferred settings here.
APP DATA\WaitZar\config.override.json.txt	Medium	Initially empty. When users change a setting via the Settings Dialog, the flat-form version of this setting is stored here. Changing this setting back to N/A will clear its entry from this file. This file is deleted if it causes any errors when starting WaitZar.
WAITZAR\config\config.json.txt	Low	Contains all “settings.*” options, see below.
WAITZAR\config\LANGUAGE\config.json.txt	Low	Specifies general settings (encodings, input methods, etc.) for a language. All languages have a sub-folder in the WAITZAR config directory.
WAITZAR\config\LANGUAGE\*\config.json.txt**	Low	Specifies arbitrary language plugins (new input methods, display methods, etc.) for a language. Plug-in developers are encouraged to keep all of their files within a sub-directory of the language that they apply to.
default-config.json.txt	N/A	This file is embedded within WaitZar.exe as a resource. If the config loading process generates an error, this file is parsed and used to configure WaitZar. It contains only the minimal settings required to load, display, and output using WaitZar and the myWin2.2 input methods.
APP DATA\WaitZar\*.bin	N/A	Various compiled keyboard files are cached here. They are named LANGUAGE.INPUTMETHOD.KBDNAME.bin.
APP DATA\WaitZar\tinysave.txt	N/A	Every time the language, input method, or output encoding is changed, these three values (and some additional data) are saved here. This is to facilitate the “Last Used” setting.

APP DATA\WaitZar\waitzar_version.txt	N/A	This is a cache of the versions file in the Google Code SVN. It is downloaded every time WaitZar starts using a background thread. This file is then read and parsed. If the version of the running WaitZar.exe isn't the first item in the list, then the user is prompted to download the newest release.
WAITZAR\config\Myanmar\WaitZar\waitzar.extrawords.txt	High	Contains custom words that are not yet in the WaitZar dictionary, or can't be added for some reason.
waitzar.userwords.txt	Low	Contains the user's list of custom words.

The important thing to remember is that files located in WAITZAR are loaded first, followed by APP DATA and then DOCUMENTS. At each stage, if a config option already exists, that option is overridden. Users should always have access to their DOCUMENTS folder, and probably APP DATA as well. Finally, if loading fails for whatever reason, then the embedded "default-config" file is used to set up WaitZar. This generally occurs either if (A) the user is running WaitZar.exe without any config directory or (B) if a developer botches the config file for his plugin.

4.2 Introduction to JSON & Config File Syntax

Config files follow the **JSON** specification, with some minor changes. In fact, WaitZar's syntax is a specialization of JSON, so it's important to understand JSON before getting into the specifics of config files. In the previous table of config files, those which follow this modified JSON syntax are colored green and bold.

4.2.1 JSON Syntax

The main JSON web site has a good description of the syntax:

http://www.json.org/

Simply put, a JSON file is a text file that contains a number of **objects** or **arrays**. An object is like a **map** or **hash table** in other programming languages. For example:

```
{
  "name" : "Aung",
  "age" : 22,
  "job" : "WZ Developer"
}
```

An array, on the other hand, is more like a **vector** or **array list**. It looks like this:

```
[1, 2, "three", 4.1]
```

Each item inside of an object or array is a **value**. Values can be strings, numbers, **null**, **true**, **false**, or another object or array. So nesting is quite common:

```
{
  "name" : "Aung",
  "past-jobs" : ["Coder", "Driver", "Chef"]
}
```

In this example we have embedded an array inside of an object's value. Note that the **key** of an object ("name" and "past-jobs" in the example above) ***must*** be a **string**. Otherwise, the format is very flexible. Check the web site for escape characters, negative numbers, and other details.

4.2.2 WaitZar's Use of JSON

WaitZar uses standard JSON with the following restrictions and additions:

- Only objects and strings are supported. (Arrays might be supported later, but are not necessary at the moment.)
  - In order to represent boolean values, we use strings. We also add "yes" as a synonym for "true" and "no" as a synonym for "false".
- Any line beginning with a hash symbol (#) is treated as a comment.
- WaitZar uses dots to transform between a "flat" and a "tree" notation.
- If an item is used for identifiation, then only alphabetical letters are considered.
  - For example, the identifiers "my-language", "mylanguage", and "MY Language" are equivalent.

The third point is the most confusing to new users. Basically, because WaitZar is guaranteed to only contain objects and strings, we allow any key at any depth within an object to be brought to the top level with dot notation. An example would help to clarify this. Consider the following configuation snippet:

```
{
  "languages" : {
    "myanmar" : {
      "encodings" : {
        "unicode" : {"display-name" : "Unicode"},
        "zawgyi-one" : {"display-name" : "Zawgyi-One"},
        "win innwa" : {"display-name" : "Win Innwa"}
      }
    }
  }
}
```

This snippet is attempting to tell WaitZar that the "Myanmar" language should have three "Encodings". This is an example of the **tree** format. To express the same information in **flat** format, simply concatenate each identifier with a period, like so:

```
{
  "languages.myanmar.encodings.unicode.display-name" : "Unicode",
  "languages.myanmar.encodings.zawgyi-one.display-name" : "Zawgyi-One",
  "languages.myanmar.encodings.win innwa.display-name" : "Win Innwa"
}
```

This syntax is much more concise, but slightly repetitive. A better option is to flatten the tree syntax only partially:

```
{
  "languages.myanmar.encodings" : {
    "unicode" : {"display-name" : "Unicode"},
    "zawgyi-one" : {"display-name" : "Zawgyi-One"},
    "win innwa" : {"display-name" : "Win Innwa"}
  }
}
```

This syntax is the most concise without requiring any repetition of information. WaitZar will transform all syntaxes into a common format when it reads in a config file.

4.2.3 Tracking Down Errors

The current version of WaitZar pre-processes all config files, so if you get a syntax error it will always be on “line 1”. When WaitZar loads an invalid config file, it will specify several letters before and after a syntax error. We intend to clean this up in future releases; for now, developers should get used to debugging JSON errors without much help from the system.

The most common errors involve adding a comma at the end of the last element in an object, or forgetting to add quotes around strings. Forgetting the opening and closing brackets are also common errors.

In general, we recommend making only small changes to config files, then restarting WaitZar and checking for errors, and then changing the config file some more. Making massive changes to multiple files all at once is a sure-fire way to get horribly confused.

4.3 Settings

(What “settings” are compared to “languages”)

4.3.1 The Settings Dialog

(quick note on how it works (stores in “local”))

4.3.2 Settings.Hotkey

(needs to be updated)

The hotkey is just another option, but changing it is the most common request we get. To do this, go to the directory you unzipped WaitZar.exe into. Look for the file “config.txt”. If it doesn’t exist, create it, open it in Notepad (*not* Wordpad) and add the line:

hotkey = Ctrl + M

This will set the hotkey to control m. If config.txt does exist, open it and look for the line:

hotkey = Ctrl + Shift

…and change it to

hotkey = Ctrl + M

…this will change the hotkey. A hotkey is written as a series of modifiers followed by a virtual key code. Modifiers include:

Ctrl

Shift

Alt

Virtual keys include any letter or number key, any modifier, and the special key **Space**. Modifiers and virtual keys are separated by plus signs. So, if you type:

hotkey = Ctrl + Alt + Shift

…then, presumably, one would have to hold "Ctrl" and "Alt" and press "Shift" to trigger Burmese. Letter values are case insensitive, so:

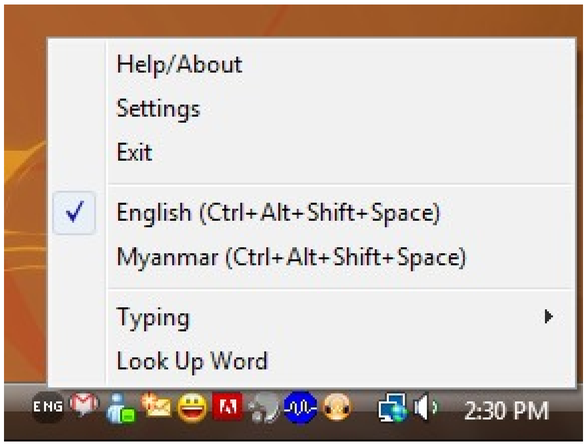
hotkey = Ctrl + M

...is the same as

hotkey = Ctrl + m

Note that hotkeys consisting of Shift plus a letter, number, or the Space bar are not allowed, since they might interfere with normal typing in WaitZar.

If you aren't sure what the current hotkey is, simply right-click on the system tray icon:


Help/About
Settings
Exit
✓ English (Ctrl+Alt+Shift+Space)
Myanmar (Ctrl+Alt+Shift+Space)
Typing
Look Up Word
ENG
2:30 PM

4.3.3 Settings.*

(needs to be updated, split into various sections. Group similar settings together)

To configure Wait Zar, you should edit the file "config.txt". This file should be present in the same directory as WaitZar.exe; if you don't have it, you can create it yourself, or just unzip it from the Wait Zar download for the current release.

The config file consists of lines structured as:

name = value

…where "name" is a single word composed of english letters, and "value" can be any number of things, depending on name. The following table lists all configurable options available at the moment. If your config.txt file does not contain a certain option, its default value will be used. If you mis-type a name, it will simply be skipped.

Please note that all **names** are *case sensitive*. If you type "Hotkey = ^m", the system will not recognize that you entered a valid option.

The following options are the most meaningful; there is a very good chance that you will have to change one of them to customize WaitZar to your needs.

Name	***Value***	***Meaning***	***Default***

hotkey	[modifier] letter, where modifier is:
	`^ for Ctrl`
	`! for Alt`
	`+ for Shift`
	…and letter is any of these, or:
	`_ for Space`
	…or any letter.
		Set the “switch language” hotkey. Use this if the default hotkey conflicts with some other task on your system. Example:
		`hotkey = ^m`
		…sets the hotkey to Ctrl+m
			^+

ballooononstart	Either of the following:
	`yes`
	`no`
		If yes, WaitZar will show a “welcome” balloon in the system tray when it first launches.	yes

trackcaret	Either of the following:
	`yes`
	`no`
		WaitZar tries to put the sentence window where you are currently typing. If you find this annoying, set this option to “no” and drag the window where you want it to stay.	yes

defaultencoding	Either of the following:
	`unicode`
	`zawgyi`
	`wininnwa`
	Also:
	`padauk`
	`parabaik`
	`myanmar3`
	…mean the same as “unicode”.
		Sets which encoding Wait Zar will use when it first loads. Set it to zawgyi or wininnwa if you use these fonts, and leave it as “Unicode” otherwise.	unicode

burmesenumerals	Either of the following:
	`yes`
	`no`
		If yes, then 0-9 will type Burmese ၀-၉.	yes

These options are less common; you should only change them if you really know what you are doing.

Name	***Value***	***Meaning***	***Default***
alwayselevate	Either of the following: `yes`	Some Vista Home Premium users will get constant errors	no

	`no`	from WaitZar. Change this to “yes”, and WaitZar will always ask you to run it as an administrator.	

lockwindows	Either of the following:
	`yes`
	`no`
		If no, the sentence window can be disconnected from the main window. Only applies if powertyping is yes.	yes

ignoremodel	Either of the following:
	`yes`
	`no`
		If yes, we do not load the embedded file Myanmar.model, and instead only load words from mywords.txt. (See Section 5.5)	no

charaset	Either of the following:
	`myanmar`
	`burmese`
	`any`
		If set to “myanmar” or “burmese”, then entries in mywords.txt must contain only Burmese letters. This allows us to use the Help keyboard even on words added by users. If set to “any”, the help keyboard is disabled, and any letter can be used for custom words. (See Section 5.5)	myanmar

fontfileregular	Any file name, or “default” or “embedded” to use the default (embedded) Zawgyi font.	Set to a file name to use this font for WaitZar’s entry window. Fonts must conform to the PulpCore standard. (See Section 5.5)	default
fontfilesmall	Any file name, or “default” or “embedded” to use the default	Set to a file name to use this font for WaitZar’s sentence window. Fonts must	Default

	(embedded) Zawgyi font.	conform to the PulpCore standard. (See Section 5.5)	

The final set of options are deprecated; they were introduced at a development crossroads some time in WaitZar's past, and they make little sense in the latest versions. Although these options are still supported, you should not change them barring exceptional circumstances —they might be removed in the future.

Name	***Value***	***Meaning***	***Default***
mywordswarning	Either of the following: `yes` `no`	User-defined words are fully-tested, but you may choose to leave this to "yes" to show a warning if a custom dictionary is used.	no
powertyping	Either of the following: `yes` `no`	If no, "space" will type each word directly, bypassing the sentence window. We highly recommend setting this to yes.	yes

4.4 Languages

(What "languages" are and what they can do)

4.4.1 Languages.*

(Overview of each languages sub-section. Explain keymagic, wordlists, encodings, etc.)

4.5 Simple Examples: Common Configuration Tasks

(Simple example stuff. Changing WZ to use a TTF font, restricting available output encodings, adding a new keyboard file.)

4.6 Complex Example: Loading a Completely Different Language

(Needs to be re-written)

WaitZar is optimized for Burmese; however, you can actually use it for any language. This is very useful if you are developing a new romanisation, and you want to test it out quickly on a real computer. It is not recommended for big languages (loading a font with every CJK ideograph will crash the system, for example). Here, we will describe how to load a custom dictionary in WaitZar.

The first step is to skip loading the Myanmar model. To do this, open config.txt and set "ignoremodel" to "yes". If you ran WaitZar now, you would not be able to type any words at all!

Next, you will want to create a new mywords.txt file that contains all of your mappings. This file should be encoded in UTF-8. Enter each word/syllable on one line, with the native word on the left and the romanised form on the right. For example:

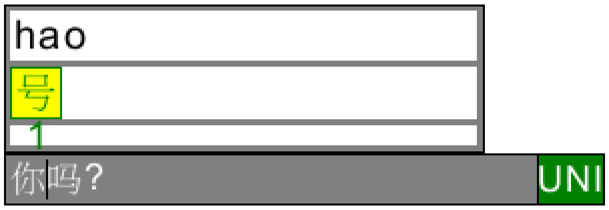
```
你好 = nihao
```

The next step is to optionally change the "charaset". Whenever possible, try to keep this set to "myanmar". For example, if you are trying to write a custom dictionary for Romabama or Burglish, you would set "charaset = myanmar". If you were trying to write one for Korean or Shan, you should set "charaset = any".

After this, try to run WaitZar. Unless you have an absurd number of dictionary entries (`sizeof(unsigned int)` on most platforms), it will start up just fine. Note that it might, however, take some time to do so.

Finally, if your romanisation does not map to the Zawgyi-One font, then you'll have to set "fontfileregular" and "fontfilesmall" to the names of the files you'll use for your language. The files "chinesefont.png" and "chinesefontsmall.png" are included with the source as examples. These are not just ordinary PNG files, however —they have several special chunks defined by PulpCore. The Java file com.waitzar.utility.PNGReCoder is included with the source to help you create this type of file. Please contact the developers if you're having trouble with this; we'd be happy to offer some advice.

Here is a screenshot of WaitZar running with a subset of the Chinese language loaded and a custom font. As you can see, it is missing a lot of important words:



5. The WaitZar Romanisation

In order to represent a script using English letters, one must define a mapping from symbols to letters —a "romanisation". Understanding this romanisation is helpful if you want to work on the WaitZar core itself. The WaitZar romanisation was designed around three simple principles:

- There should be a few simple rules which most words follow.
- All words should be easy to guess, even if you don't know the rules.
- Each Burmese word gets exactly ***one*** English romanisation.

The first rule is covered in the next section (on the default romanisation), and the second rule is relevant throughout. The final rule may seem silly: why not allow တာ = “tar” or “thar” or “ttar”? Our main reason for enforcing this rule is efficiency. Once you learn that သာ = “thar” in WaitZar, then you will start to get annoyed if you see သာ when you type “tar” or “ttar”. Through decisions like this, we try to make our system agreeable to novices and experts alike.

5.1 Default Romanisation

Most Burmese words can be thought of as an onset combined with a rhyme. For example: နုံ, pronounced “non”, can be broken up into န = “n(a)” and --ုံ = “on”. Words can actually be broken up further, into consonants, medials, vowels, tones, and finals, but a unique quality of onsets and rhymes is that they can be spelled the same way when they appear in different words. For example:

န → n		--ုံ → on	
Myanmar	*Roman*	*Myanmar*	*Roman*
နုံ	non	နုံ	non
နင်	nin	ခုံ	khon
နီ	ne	စုံ	son
နေ	nay	ရုံ	yon
နွယ်	nwal	သုံ	thon

The rhymes always follow their phonetic pronunciation; the onsets follow a usage-based taxonomy. The following rules apply to onsets:

1. The first sound of the Myanmar letter is the Roman letter.
2. If two onsets share the same first sound, the less common onset has an “h” after it. (For example: ဒ is “d”, but ဓ is “dh”)
3. Rarely-used letters won't add an “h”. (For example: ဏ and ဉ are both “n”, even though န is also “n”).

For reference, here is the current list of rhymes and onsets. Within a rhyme, a dash indicates where onsets may be plugged in. You will need the Zawgyi-One font to view these tables.

Onsets

Myanmar	*Roman*	*Myanmar*	*Roman*	*Myanmar*	*Roman*	*Myanmar*	*Roman*
က	k	ဋ	tt	ဓ	dh	ဝ	w
ခ	kh	ဍ	dh	န	n	သ	th

ဂ	g	ဌ	ht	ပ	p	ဟ	h
ဃ	gh	ဋ္ဌ	ht	ဖ	ph	အ	
င	ng	ဏ္ဍ	n	ဗ	v	ဥ	o
စ	s	ဍ	d	ဘ	b	ရှ	sh
ဆ	ss	ဏ	nh	မ	m	သျှ	sh
ဇ	z	တ	t	ယ	yh	လျှ	sh
ဈ	zh	ထ	ht	ရ	y	ရွှ	shw
ည	ny	ဒ	d	လ	l		

Rhymes

my	*rom*	*my*	*rom*	*my*	*rom*	*my*	*rom*	*my*	*rom*	*my*	*rom*
-	a	-ုဋ်	ote	-ျာ့	ya	ေ-ြာ့	ywae	-ွိုင်	oi	ေ-့ရှ	shae
-ာ/-ါ	ar	ေ-ဋ်	ate	-ျား	yar	ေ-ြာ်	yaw	-ွစ်	wet	-ွှ	ywa
-ာ့/-ါ့	a	ေ-ာဋ်	at	-ျိ	yi	-ြံ	yan	-ွဋ်	ut	-ွှနုပ်	yaunote
-ား/-ါး	arr	-ါဌ	at	-ျီ	ye	-ြံ့	yant	-ွတ်	oot	ေ-ွှ	way
-ိ	i	-ဏ်	an	-ျီး	yee	-ြို	yo	-ွန်	on	ေ-ွှး	way
-ီ	e	-ာဏ်	am	-ျု	yu	-ြို့	yoe	-ွန့်	ont	-ွှဲ	wal
-ီ့	eet	-ိဏ်	ai	-ျူ	yuu	-ြိုး	yoe	-ွန်း	onn	-ွှံ	yon
-ီး	ee	-ုဏ်	ohn	-ျူး	yuu	-ြုံ	yone	-ွပ်	wot	-ွှက်	yot
-ု/-ု	u	-ိုဏ်	hite	ေ-ျ	yay	-ြုံ့	yote	-ွမ်	wan	-ွှင်	ywin
-ူ/-ူ	uu	-ိုဏ်း	ine	ေ-ျး	yay	-ြုံး	yone	-ွမ်း	wann	-ွှတ်	yut
-ှူ	hu	-တ်	at	-ျဲ	yal	-ြက်	yat	-ွယ်	wal	-ွှန်	hon
-ူး/-ူး	uu	-တ်စ်	at	-ျဲ့	yae	ေ-ြာက်	yout	-ွယ့်	wae	-ွှန်း	honn
ေ-	ay	-ာတ်/-ါတ်	at	ေ-ျာ	yaw	-ြိုက်	yite	-ှ	ha	-ွှမ်း	yun
ေ-့	ae	-ိတ်	ate	ေ-ျာ့	yut	-ြင်	yin	-ှာ	har	-ျှ	hwa
ေ-း	ay	-ုတ်	ote	ေ-ျာ်	yaw	-ြင့်	yint	-ှာ့	ha	-ျှား	lyar
-ဲ	al	-ူတ်	hote	-ျံ	yan	-ြင်း	yinn	-ှား	har	ေ-ျှာ	haw
-ဲ့	ae	ေ-တ်	hit	-ျံ့	yant	ေ-ြာင်	yaung	-ှိ	eet	ေ-ျှာ့	hawe
ေ-ာ/ ေ-ါ	aww	ေ-ါ့တ်	ot	-ျို	yo	ေ-ြာင့်	yout	-ှီ	he	ေ-ျှာ်	haw
ေ-ာ့/ ေ-ါ့	ot	-ဒ်	at	-ျို့	yoe	ေ-ြာင်း	yaung	-ှီး	hee	-ျှို့	hloe

ေ-ာ်/ေ-ါ်	aw	-ာဒ်/-ါဒ်	at	-ျိုး	yoe	ြ-ိုင်	yai	-ှူ	uu	-ျှင်	hlin
ေ-ာ့	ot	-ိဒ်	kit	-ျုံ	hone	ြ-ိုင်း	yine	-ှု့	hu	-ျှစ်	hit
-ံ	an	-ုဒ်	oat	-ျုံ့	hote	ြ-စ်	yit	-ှူး	huu	-ျှဉ်း	yinn
-ံ့	ant	-ိုဒ်	ite	-ျုံး	yone	ြ-ိစ်	yit	-ှူ/-ှူ	huu	ြ-ွ	wa
-ို/-ို	o	-န်	an	-ျက်	yat	ြ-ဉ်	yin	-ှူး/-ှူး	huu	ြ-ွာ	war
-ို့/-ို့	hoet	-န့်	ant	ေ-ျာက်	yout	ြ-ဉ်း	yinn	ေ-ှ	hay	ြ-ွား	wyar
-ိုး/-ိုး	oe	-န်း	am	-ျိုက်	yite	ြ-ည်	yi	ေ-ှ့	hae	ေြ-ွ	way
-ုံ/-ုံ	one	-ာန်	han	-ျဂ်	at	ြ-ည့်	yae	ေ-ှး	aye	ေြ-ွ့	ywae
-ုံ့	ote	-ါန်း	ann	-ျင်	yin	ြ-ည်း	yee	-ှဲ	hal	ေြ-ွး	way
-ုံး	one	-ိန်	ane	-ျင့်	yint	ြ-ဋ်	yat	-ှဲ့	hlae	ြ-ွက်	wyat
-ှိုး	hoe	-ိန့်	ate	-ျင်း	yin	ြ-တ်	yat	ေ-ှာ	aww	ြ-ွင်း	wyin
-က်	at	-ိန်း	ane	ေ-ျာင်	aung	ြ-ိတ်	yate	ေ-ှာ်	haw	ြ-ွတ်	yoot
-ိက်	ate	-ုန်	one	ေ-ျာင်း	yaung	ြ-ုတ်	yote	-ှံ	han	ြ-ွန်	yoon
-ုက်/-ုက်	ote	-ုန့်	oat	-ျိုင်	hai	ြ-န်	yan	-ှံ့	hant	ြ-ွပ်	yut
ေ-ာက်/ေ-ါက်	out	-ုန်း	one	-ျိုင့်	yite	ြ-န့်	yant	-ှို	ho	ြ-ွမ်း	wyan
-ိုက်	ite	-ပ်	at	-ျိုင်း	yine	ြ-န်း	yan	-ှို့	hoet	ြ-ွယ်	ywal
-ှိုက်	ite	-ိပ်	ate	-ျစ်	yit	ြ-ိန်	yai	-ှိုး/-ှိုး	hoe	ြ-ှူ	yu
-ုဒ်	hote	-ုပ်	ote	-ျဉ်	yin	ြ-ိန့်	hyate	-ှုံ	hone	ေြ-ှး	yae
-ဂ်	at	ေ-ာပ်	at	-ျဉ်း	yin	ြ-ိန်း	yane	-ှုံ့	hote	ြ-ှုံး	yone
-ိဂ်	ate	-ဗ်	at	-ျာဉ်	in	ြ-ုန်း	yone	-ှုံး	hone	ေြ-ှာက်	yout
-ုဂ်	oat	-ိဗ်	ait	-ျည်	yi	ြ-ပ်	yat	-ှက်	hat	ြ-ှိုက်	yite
-င်	in	-ုဗ်	ote	-ျည့်	yeet	ြ-ုပ်	yote	ေ-ှာက်	hout	ြ-ှင့်	yint
-င့်	int	-ာဘ်	at	-ျည်း	yee	ြ-ုဗ်	yote	-ှိုက်	hite	ေြ-ှာင်	yaung
-င်း	inn	-မ်	an	-ျတ်	yat	ြ-မ်	yan	-ှင်	hin	ေြ-ှာင့်	yout
-ိင်	ain	-မ့်	amt	-ျိတ်	yate	ြ-မ်း	yan	-ှင့်	hint	ေြ-ှာင်း	yaung
ေ-ာင်/ေ-ါင်	aung	-ီမ့်	eint	-ျုတ်	yote	ြ-ိမ်	yane	-ှင်း	hinn	ြ-ှပ်	hote
ေ-ာင့်/ေ-ါင့်	out	-မ်း	ann	-ျန်	yan	ြ-ိမ့်	yate	-ှိင်	hane	ြ-ှုပ်	yote
ေ-ာင်း/ေ-ါင်း	aung	-ိမ်	ain	-ျန့်	ant	ြ-ိမ်း	yane	ေ-ှာင်	haung	ြ-ှိမ်း	yine
-ိုင်	ine	-ိမ့်	aint	-ျန်း	yan	ြ-ယ်	yal	ေ-ှာင့်	hout	-ွှ	lwa
-ိုင့်	hite	-ိမ်း	ane	-ျိန်	yane	ြ-ိုလ်	hlo	ေ-ှာင်း	haung	-ွှာ	hwar
-ိုင်း	ine	-ုမ်	one	-ျိန်း	yane	ြ-သ်	at	-ှိုင်	laing	-ွှား	war
-ှိုင်	aing	-ုမ်း	hone	-ျုန်	yone	ြ-ိသ်	hate	-ှိုင်း	laing	-ွှီး	wee

-စ်	it	-ယ်	al	-ျုန်း	yone	-ြပ်	at	-ှစ်	hit	ေ-ွှ	lwae
-ာစ်	it	-ယ်လ်	al	-ျပ်	yat	-ြာပ်	at	-ှဉ်	hin	ေ-ွှ့	lwae
-ားစ်	hars	-ယ့်	ae	-ျိပ်	yate	-ြိုပ်	ho	-ှဉ်း	hin	ေ-ွှး	wae
-ိစ်	ate	-ာယ်/-ါယ်	hal	-ျုပ်	yote	-ွ	wa	-ှည့်	lae	-ွှဲ	wal
-ုစ်	hit	-ိုယ်	o	-ျူပ်	yote	-ွာ/-ွါ	war	-ှည်း	hal	-ွှံ	won
ေ-ာ့စ်	oss	-ိုယ့်	oe	-ျမ်း	yan	-ွာ့	wa	-ှတ်	hat	-ွှံ့	yon
-ိုစ်	ost	-ရ်	al	-ျိမ့်	aint	-ွား	warr	-ှိတ်	hate	-ွှင်	hwin
-ဇ်	it	-ာရ်	an	-ျယ်	yal	-ွိ	weet	-ှုတ်	hote	-ွှင့်	wint
-ာဇ်	yiz	ေ-ရ်	hay	-ြ	ya	-ွိ့	weet	-ှန်	han	-ွှင်း	hwin
-ိဇ်	ist	-ိုရ်	ho	-ြာ	yar	-ွီ	wee	-ှန့်	lant	-ွှတ်	oot
-ုဇ်	yit	-လ်	Al	-ြား	yar	-ွီး	hee	-ှန်း	han	-ွှန်	yon
-ဉ်	in	-ိလ်	ain	-ြိ	yi	ေ-ွ	way	-ှိန်	hane	-ွှန်း	yon
-ဉ့်	int	-ုလ်	ho	-ြီ	ye	ေ-ွ့	wae	-ှိန်း	hane	-ွှပ်	wat
-ဉ်း	inn	ေ-ာလ်	all	-ြီး	yee	ေ-ွး	way	-ှုန်	hone	-ွှမ်း	wan
-ာဉ်	im	ေ-ာ်လ်	all	-ြု	yu	-ွဲ	wal	-ှုန့်	hote	-ျွှန်	yun
-ိဉ်	yin	-ိုလ်	ho	-ြူ	yu	-ွဲ့	wae	-ှုန်း	hone		
-ုဉ်း	hone	-သ်	at	-ြူး	yuu	-ွံ	on	-ှပ်	hat		
ေ-ဉ်	ay	-ိသ်	ate	ေ-ြ	yay	-ွံ့	ont	-ှိပ်	hate		
-ည်	i	-ုသ်	ote	ေ-ြ့	hae	-ွက်	wat	-ှုပ်	hote		
-ည့်	eet	ေ-သ်	aye	ေ-ြး	way	-ွိုက်	wite	-ှမ်း	hann		
-ည်း	ee	-ိဟ်	ai	-ြဲ	yal	-ွင်	win	-ှိမ်	hane		
-ဋ်	oot	-ျ	ya	-ြဲ့	hae	-ွင့်	wint	-ှိမ့်	ate		
-ိဋ်	ate	-ျာ	yar	ေ-ြာ	yaw	-ွင်း	win	-ှယ်	hal		

Warning: You may have noticed that “အ” is *not* romanised as “a”. This is because it functions as a silent, “placeholder” onset in spoken Burmese. By way of example, “အုတ်” should not be romanised to “aote”, but simply “ote”.

Bigger Warning: Please be careful not to read the rhymes with an English pronunciation. For example, “အုန်” as “one” should *not* be read to sound like the number 1 in English. If you consider another word like “ကုန်” as “kone”, you won’t run into this problem.

5.2 Special Romanisation

Some words don’t fall into any simple pattern, and you’ll just have to memorize them. These include pat-sint and foreign words. Also, there are six words that directly contradict our onset + rhyme rule; these include “လျှို့”, which we represent as “lhyo” instead of “sho”. The following table lists (in Zawgyi-One) these six words, and the pat-sint words. For brevity, we do not include foreign words, which romanise to their foreign spelling. Please note that pat-sint words can also be entered syllable-wise, which many beginners find easier to type. Section 3.3 describes how to do this.

Special Words							
Over-rides		ဝန္တာ	wontar	ရစ္စ	yitsa	နန္ဒ	nanda
လျှို့	lhyo	ဂုတ္တ	goatta	ကစ္စ	kitsa	စန္ဒ	sanda
လျှိုး	lhyoe	သန္တာ	thandar	နစ္စ	nitsa	ဆန္ဒ	sanda
လျှက်	lhyat	သန္တ	thanda	မစ္ဆ	myitzi	အိန္ဒိ	indi
လျှင်	lhyin	အတ္တ	atta	္ဝ		ကိန္ဒ	kaida
လျှပ်	lhyat	လန္တိတ်	lantate	သိက္ခာ	theitkhao	အိန္ဒြေ	eidray
လျှမ်း	lhyan	မုတ္တ	moteta	္ခ		အိန္ဒြော	eidra
္ဘ		ရိတ္တ	yateta	ဒုက္ခ	dotka	နန္ဒာ	nandar
ကမ္ဘာ့	kaba	္ဓ		ဒုက္ခိ	dotkhe	မုဒ္ဒ	moteda
ကမ္ဘ	kamba	ခန္ဓာ	khandar	ယက္ခ	yatkha	ဣန္ဒြေ	eaidray
ကမ္ဘာ	kabar	ဗန္ဓု	bandu	ရုက္ခ	ratka	္ဂ	
ကမ္ဘော	kambaw	ဗုဒ္ဓ	buddha	လက္ခ	latkha	ဘဂ္ဂ	batga
္က		္မ		သိက္ခာ	theitkhar	ပုဂ္ဂိုလ်	poatgo
စက္ကန့်	setkant	ဓမ္မ	damma	ရက္ခိ	yatkhe	မဂ္ဂ	matga
စက္ကူ	satku	ဓမ္မာ	dammar	စက္ခု	satkhu	ပုဂ္ဂ	poatga
မက္က	matka	ကြမ္မာ	kyanmar	င်္		္ဏ	
ကျိုက္က	kyiteka	သမ္မာ	tanmar	အင်္ဂ	inga	ပုဏ္ဏား	ponenar
ဥက္က	oatka	သမ္မ	tanma	အင်္ဂါ	ingar	ဏ္ဍ	
တက္က	tatka	ရမ္မက်	yatmat	ဒင်္ဂါး	dingar	ဘဏ္ဍာ	bandar
သက္က	thatka	္ထ		နင်္ဂ	ninga	သဏ္ဌာန်	thatan
္လ		ဝတ္ထု	woothtu	မင်္ဂ	minga	သဏ္ဍာန်	thandan
မိလ္လာ	mailar	မိတ္ထီ	matehti	သင်္ကေ	tinkay	ဒဏ္ဍာ	dantar

◌္ဍ		◌္ဆ		ဒင်္ဂ	dinga	ကဏ္ဍ	ganda
သဏ္ဍာန်	thandan	ပုစ္ဆာ	poatsar	သင်္ကန်း	thingan	***Misc***	
ပဏ္ဍာ	patnar	ဣစ္ဆိ	eightsi	ဟင်္သာ	hintar	ဪ	aww
ကဏ္ဍ	ganda	မစ္ဆိ	myitseet	နင်္လာ	ninlar	ဩ	aww
◌္တ		ရစ္ဆာန်	yutesan	ဘင်္ဂ	binga	သြ	aww
စန္တ	santa	င်္◌		သင်္ကြန်	thingyam	သို့	the
အန္တ	anda	သင်္ချိုင်း	thingjine	သင်္ဘော	thinbaw	လကျာ်	latyar
နတ္တိ	nethti	င်္◌		သင်္ချာ	tinchar	ဧည့်	ae
ကတ္တာ	kattar	အင်္ကျီ	inngi	စင်္ကာပူ	singapore	ဧ	ae
ပြိတ္တာ	pyittar	◌္ဈ		အင်္ဂလိပ်	english	ဤ	ei
မတ္တီ	matedhi	သဇ္ဈာ	tinzar	◌္ပ		၏	eat
ဝတ္တ	wootta	ဝိဇ္ဈာ	waitzar	သိပ္ပံ	theitpan	၍	ywae
မေတ္တာ	myittar	◌္စ		သမ္ပာ	thanpar	၌	nhite
ခေတ္တ	khitta	ပစ္စည်း	pyitsee	သပ္ပာယ်	thetpal	ဦ	ou
ပုတ္တ	poatta	သစ္စာ	thitsar	ကပ္ပ	katpa	ဦး	oo
ပတ္တ	patta	ဥစ္စာ	oatsar	ကုမ္ပ	konepa	၎င်း	lakaung
သတ္တာ	tattar	ဝစ္စ	wootsa	ဓိပ္ပါယ်	datepal	ဿ	tha
သတ္တု	thattu	ပစ္စ	pyitsa	စမ္ပော	sanpaw	ယောက်ျား	youtkyar
သတ္တိ	thetthe	ကိစ္စ	katesa	စမ္ပာ	sanpar		
အန္တာ	antar	စစ္စ	sitsa	သပ္ပါယ်	thatnal		
မန္တ	manda	အစ္စ	itsa	◌္ဗ			

5.3 Shortcuts

Two typing shortcuts are hardcoded into WaitZar's representation scheme:

- Any word containing "aung" can shorten it to simply "g". For example, "kaung" can be typed as "kg". "Aung" by itself can be shortened to "ag".
- If only one set of words is possible at any point, the sequence to that set is parenthesized. For example, typing "singa" is sufficient for WaitZar to guess that you want "singapore".

5.4 Dictionary Lookup

If, for any reason, you cannot find the word you want to type in WaitZar, you can search for it in the generated wordlist:

http://waitzar.googlecode.com/svn/trunk/eclipse_project/wordlist.generated.txt

There is also the Help Keyboard; see the User's Guide, section 4 for details.

5.5 Known Font Issues

This section requires the Zawgyi-One font to render properly.

Unicode 5.1 is still somewhat new, and there are a few glitches in fonts which support the latest standard. When possible, Wait Zar will re-organize the letters in a word to hide these errors. However, this is not always possible. Please be aware of these errors. If you are a font developer, we'd love to hear from you —give us an email and we'll offer whatever suggestions we can.

We apologize if any of these words are considered "incorrect" by a font developer; we will remove them from the dictionary if you can give us sufficient examples of words to use in their place. (We have already done this for several words, after excellent feedback from the Padauk developers).

Note that Parabaik is no longer tracked, as it hasn't been updated in several years. We'll add it back to this table if it releases an update.

Sequence (Zawgyi-One)	**Encoding** (Unicode 5.1)	**Problem**	**Solved?**
~~ယောင်္ကျား~~	\u101A\u1031\u102C \u1004\u103A\u1039 \u1000\u103B\u102C \u1038	N/A	Removed from WaitZar; original word was a misspelling.
လကျာ် (slang term)	\u101C\u1000\u103B \u102C\u103A	Myanmar3 cannot attach "်" correctly to "ာ".	Requires font modification.
ဆွိ့	\u1006\u103D\u102D \u1037	Myanmar3 cannot attach "့" correctly to "ွ". Switching "ွ" and "ိ" doesn't help.	Requires font modification.
ညှိုး	\u100A\u103E\u102D \u102F\u1038	WaitZar contains both words (short/tall leg). Myanmar3/Ayar/Yunghkio show tall "ု". Padauk shows short "ု".	In progress. *Requires change to the WaitZar romanisation.*
ညု	\u100A\u102F	Padauk/Ayar display this	Requires font modification.

		(in any form) with short "ု". Yunghkio/Myanmar3 show tall "ု".	
နိူက် Also occurs for: နိူင်	\u1014\u102D\u1030 \u1000\u103A	Myanmar3 and Padauk do not render "ူ" in the right location. Myanmar3, & Yunghkio all fail to shorten "န" to its short form "န". Switching "ူ" and "ိ" doesn't help.	Requires font modification.
မှူး	\u1019\u103E\u1030 \u1038	WaitZar contains both words (short/tall leg). Myanmar3 and Yunghkio render short "ူ" as tall "ူ". Switching the encoding order doesn't help.	In progress. *Requires change to the WaitZar romanisation.*
မိုူး (this word seems odd, syntactically)	\u1019\u102D\u102F \u1030\u1038	Myanmar3 encodes tall "ူ" as short "ူ". Switching the encoding order doesn't help .	In progress. *Possible change to the WaitZar romanisation.*
ရှူ and ရှူး	\u101B\u103E\u1030 (opt:\u1038)	Padauk converts tall "ူ" to short "ူ".	Requires font modification.
လှူ	\u101C\u103E\u1030	Padauk & Ayar convert tall "ူ" to short "ူ".	Requires font modification.

6. Coder's Guide

(Stuff on getting the source, compiling, general way the program works, etc.)

6.1 Getting the Source

(description of Google Code, SVN, maybe TortoiseSVN too.)

6.2 Compiling and Running Wait Zar from Source

Wait Zar is written in C++, and it makes use of many features from the latest standard (C++0x). Thus, you will need a compiler that can support these. Currently, we recommend using the Eclipse CDT and MinGW. You can download them here:

http://www.eclipse.org/cdt/

http://www.mingw.org/

Install MinGW first. You should get the **Automated MinGW Installer**, which will be titled something like “**mingw-get-inst-XXX.exe**”, where “XXX” is a version number.

You should choose to “Download latest repository catalogues” when prompted to do so. The installer will recommend installing in “C:\MinGW”, which is fine. When you are asked to “Select Components”, you should choose the **C++ Compiler**, the **MSYS Basic System**, and the **MinGW Developer ToolKit**. (Note: I'm not sure about the last two. Can't hurt to have them, though.)

For Eclipse CDT, you may either download the IDE from the link provided above, or you may upgrade an existing installation of Eclipse using the “Install New Software...” feature from the IDE. You will have to enter a repository URL. Consult the download page; for Eclipse Helios, it will be something like this:

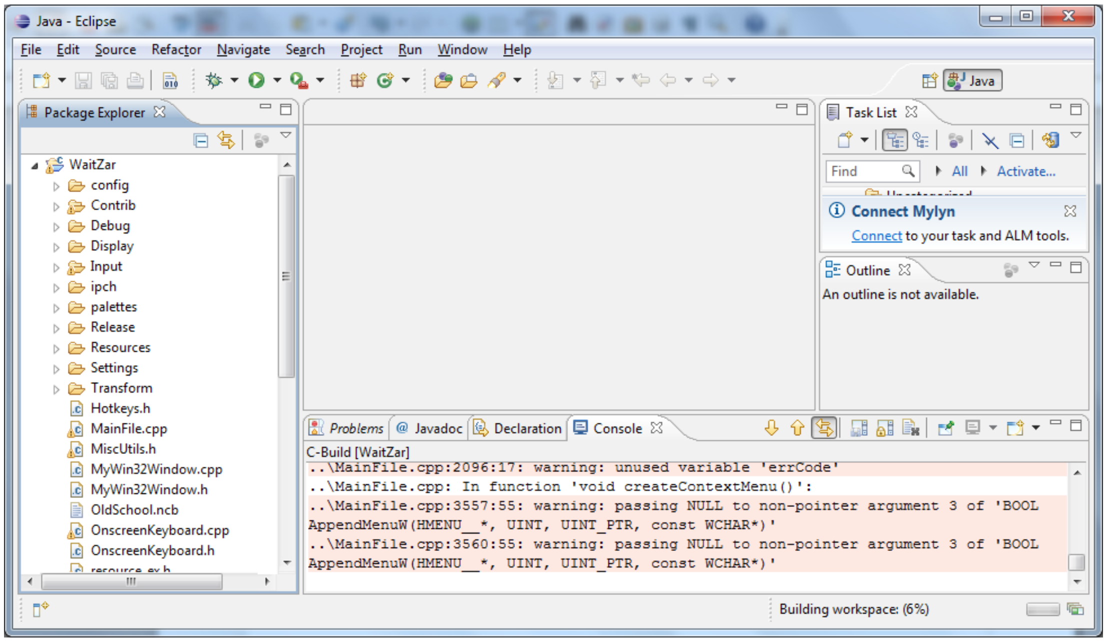
http://download.eclipse.org/tools/cdt/releases/helios

When you start Eclipse for the first time, it will ask you to choose a Workspace. This won't affect anything, so just click “Use this as the default and do not ask again”. Finally, depending on how you install the Eclipse CDT, you may be asked to pick various “optional” modules. You can choose “C/C++ GDB Hardware Debugging” if you want support for debugging Wait Zar.

You are now ready to compile Wait Zar.

6.2.1 Configuring the Eclipse CDT

Start Eclipse, and close the Welcome screen that appears. Now, choose “File -> Import...”. Expand the “General” tab and choose “Existing Projects into Workspace”. Hit “Next”. Browse to the **win32_src** directory inside the main Wait Zar source directory. Make sure the “Wait Zar” project is selected, then choose “Finish”. You will now see Wait Zar displayed in the package explorer, and the Console will show that a build has begun.



```
C-Build [WaitZar]
..\MainFile.cpp:2096:17: warning: unused variable 'errCode'
..\MainFile.cpp: In function 'void createContextMenu()':
..\MainFile.cpp:3557:55: warning: passing NULL to non-pointer argument 3 of 'BOOL
AppendMenuW(HMENU__*, UINT, UINT_PTR, const WCHAR*)'
..\MainFile.cpp:3560:55: warning: passing NULL to non-pointer argument 3 of 'BOOL
AppendMenuW(HMENU__*, UINT, UINT_PTR, const WCHAR*)'
```

You should wait until the Build completes. At this point, you should probably learn about the different build options. Wait Zar currently supports two build profiles: Debug and Release. The first of these generates a large (60 MB) executable with lots of debugging information and no optimization. You can run this from Eclipse with the Debugger if you are trying to fix bugs in the program. The second, however, takes longer to compile and generates a smaller (2 MB) faster executable. You can change configurations by right-clicking on the “Wait Zar” project in the Package Explorer, choosing “Build Configurations”, then “Set Active”, and finally, the configuration you want active.

Make sure you click “Project” and then uncheck “Build Automatically”. Otherwise, Eclipse will try to re-build the project after every major change. Since building can take 4 or 5 minutes, this will end up wasting a lot of time if you do not uncheck it.

6.2.2 Understanding the Different Configurations

(Brief overview of Debug versus Release build. Speed of EXE. Time to compile. Debugging support (vector bounds checks), etc. Note: Might have to change this to “compiler options”, etc.)

6.2.3 Compiling

(General build steps. Note about errors (top-most are the most relevant). How to force a complete rebuild (delete “Debug” and “Release” directories). Note that for some constants (e.g., logging ones) this seems to be required.)

6.2.4 Running

(Normal running. Debugging live. The “working directory” (win32_source) and why you can't just double-click on it from “Release”. How to copy a build to the desktop & run it correctly (copy word list & config dir. too) Running a test file. (-t).)

6.3 Understanding the Code

(Overall description; it started with just Burmese. Stuff added on; can sometimes be difficult to understand. Also win32 can be tricky. Generally single-threaded, etc..)

6.3.1 Startup Events

(Go through a log file, describe what happens. Reference the source. Screenshots. Explain Window Classes briefly.)

6.3.2 Bundled Resources

(Note how in Windows, one generally bundles critical resources inside the EXE. Explain how a “resource” can be just a file. Maybe extract resources and look at them manually? Explain how menus and dialog boxes are loaded from resources.)

6.3.3 Loading a Configuration

(How the various config files are loaded and combined into one. Note about “lastused” hack. Fallback to the default config.)

6.3.4 Flow of Input

(Describe hotkeys. Explain how input is passed to the “InputMethod” which might also pass it along to other things. Explain when and why windows are recalculated, hotkeys are turned on and off, etc.)

6.3.5 The Help Keyboard

(Logical extension of “Flow of Input”. This uses a LetterInputMethod, and just manipulates the strings a bit. Note how eventually we want to be able to use any Letter method for the Help Keyboard. Note how we can “click” on areas of the window.)

6.3.6 The System Tray Icon and Menus

(How we build “Owner Drawn” menus for languages, etc. How we embed Pdk-Zg. General other stuff related to the system icon.)

6.3.7 Miscellaneous Interesting Classes

(MyWin32Window, Logger, others?)

6.3.8 Design Philosophy

(Link to the text document.)

Appendix A: License

Wait Zar is licensed under the Apache License 2.0; see the file LICENSE for more information.

This means that Wait Zar is open source, so you can always modify it and redistribute it as long as your comply with the license. For example, if you wanted to use the code to write a MacOS plugin, you could do that. Of course, we'd like to hear about all projects like this, so please email **help@waitzar.com** if you're working on something that uses the Wait Zar library. We're happy to help answer your questions and test your project.

Some of the additional parts of Wait Zar's functionality are provided under different (but compatible) licenses such as the GPL and Creative Commons. If you'd like to contribute to Wait Zar, but you want to do so under a different license, please let us know and we'll arrange for it (as long as the licenses are compatible with the Apace License 2.0).