MINISTRY OF EDECATION & TRAINING

VIET NAM POSTS AND TELECOMMUNICATIONS GROUP

POSTS AND TELECOMMUNICATIONS INSTITUTES OF TECHNOLOGY

GRADUATED DESIGN

OBJECT NAME: Develop the Lao Input Method (LaoKa)

STUDENT: SOMNUC ANOUSINH.

COURSE: 2004 - 2009

HANOI – 11/2006

MINISTRY OF EDECATION & TRAINING

TẬP ĐOÀN BƯU CHÍNH VIỄN THÔNG VIỆT NAM

POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY

GRADUATED DESIGN

OBJECT NAME: Develop the Lao Input Method (LaoKa)

Start Date:
Finish Date:

Instructor: PhD. Ha Hai Nam
Student: Somnuc Anousinh

HANOI - 2006

TABLE OF CONTENT

ABSTRACT

Nowadays, commercial economic services grow very fast and move towards globalization tendency, so exchanging information becomes extremely important among nations all over the world. In spite of many benefits, people are not only interested in but also worried about it. As the results, in these practical conditions, everyone has to rise up productivity. Everybody must have ability in solving many works within fixed and limited time which keep us up direct appointments because of geographic boundary. So communication via the mobile phone, chat, mail becomes important and popular, of which mail and chat are preferable. Along with the global development, Laos is a developing country where the demand of exchanging information, particularly via chat and email, both national and international, is very high. In connection with Lao script which have 56 different letters, as much as two times of the number of the Latin letters and it is common knowledge that keyboard support chiefly to enter the letters. If we want to input information with language system without using Latin letters like Lao language, we have to assign Laos characters on current letters in keyboard which are definitely complicated and difficult for end user. As a result, developing a Lao input method by using Latin letters to compile and display Lao letters is easier for inputting information and typing on keyboard is always a ceaselessly works because practical demand change incessantly. To satisfy this practical condition and contribute my efforts into the development a convenient typing method using the current common keyboard for my country as well as technology, being a student studying the information of technology, I decide to apply what I have been taught at the class into this hard work. I strongly believe that Laos input method will be an useful and helpful tool to assist people using computer to type Laos language much easier.

ACKNOWLEDGEMENT

First of all, I would like to say thank sincerely to PhD. Ha Hai Nam for his kind assistance in providing data, directing and correcting me during the hard time to complete graduated design.

Second, I also would like to express heartfelt thank to the lecturers at Information of Technology Department who not only taught me dedicatedly technical knowledge but also gave me a piece of lifestyle and professional etiquette which help me much in getting over difficulty in language culture lifestyle in university environment in Vietnam. The teachers also advise me on my own working plan after the graduation and show me the way I must walk on.

Third, I would like to give thanks to my colleagues studying at Information of Technology Department. During more than four years, they gave me hand to overcome troubles in communication and language. They instructed knowledge on class, explained in details for Vietnamese words that I did not understand. Especially, through the process of preparing graduated design, they are spending much time to share their value comments to my subject and help me to finalize it.

Finally, I thank greatly all teachers and colleagues at the Posts and Telecommunications Institute of Technology who make favor condition for Laos students in general and particularly for me by giving a good environment to study and live that make me feel at home.

1. STATE OF THE ART OF LAO INPUT METHOD.

Some servicing providers early recognized Laos using demand on the computer. To apply this demand, two providers gave two differences Lao Input Method that have been using at the present time. Now, we will spend a bit of time to learn about us (LaosScript and LaosUnikey Input Method).

1.1. LaosScript.

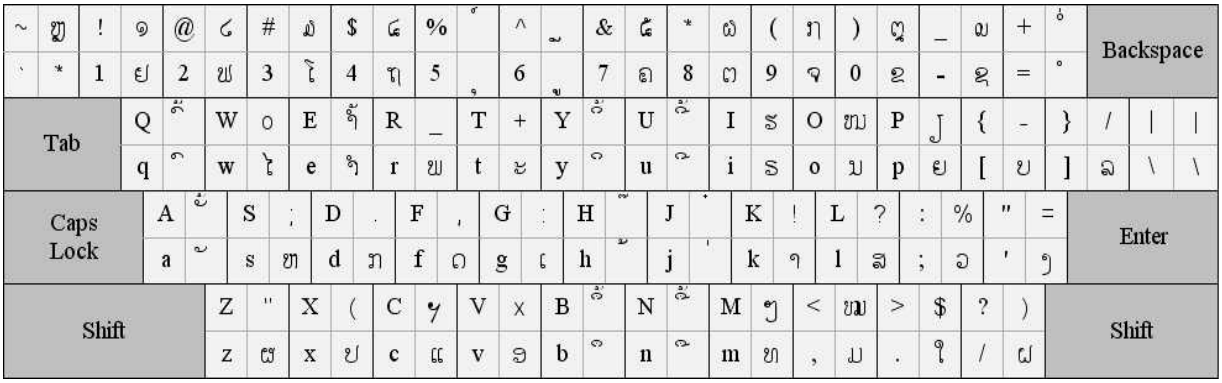
This LaosSript was developed by Dr. John. M. Durdin who gave LSWIN standard and Laos standard 2005 with various fonts such as LSWIN, Unicode, Laos 95, Laos 2000, IBM STENNO, LAOS ISO. Its head office is in American.

LaosScript assists Lao language using to become easier in various window applications. So it's called LSWIN (LaosScript for window). Lao Script for Windows has been widely used in Lao PDR and around the world since its first release in 1993.

Lao Script for Windows includes both Unicode and non-Unicode Lao fonts, and a keyboard mapping application that automatically wraps Lao text at word boundaries with compatible applications.

Mapping rule among keyboard characters with Lao letter is recognized by images follow:

LSWIN:



1.2. Laos Unikey.

Laos Unikey was developed by Laosoftware team; its head office is in France. Laos Unikey includes various Lao standards such as Duang Chan (LSWIN), Lao US (French Keyboard), Lao France, Lao France New, Sida Thong. Defining Lao standards was also base on mapping characters from the keyboard.

Mapping rule among keyboard characters with Lao letter is recognized by images follow:

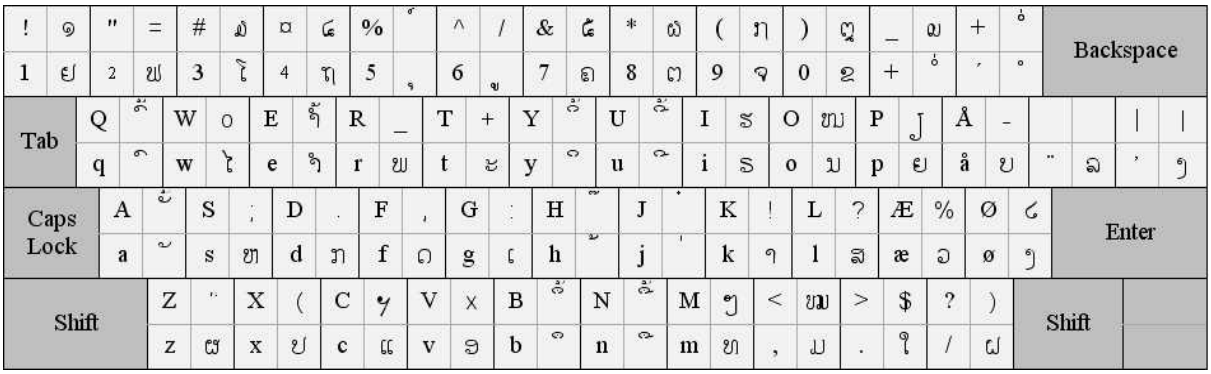
1.2.1. Duang Chan.

1.2.2. Lao US.

1.2.3. Lao France.

1.2.4. Lao France New.

1.2.5. Sida Thong.



1.3. Advantages and disadvantages of LaoScript and Lao Unikey.

1.3.1. Advantage

Assist for instant communication between Lao people via application on the Internet and edit text by Lao easily.

This is the first step marking success in taking the Lao language using into computer applications

1.3.2. Disadvantage

The above introduction and images about mapping rule among keyboard characters with Lao letter. LaosScript and LaoUnikey reveal some disadvantage follow:

We know that Lao writing language have many letters (more than 56 letters) while keys is limited on the keyboard according to the international standard. So, that mapping rule request user to remember key and key combination when typing the Lao letters. This makes difficulty in editing text as well as wasting much time.

2. LAOKA.

2.1. Ideal and initiative.

Nowadays, in practical conditions, most people using the Lao language to communicate via the Internet such as chat, email, etc… they must use Latin letter due to Lao's phonetic symbol. Although typing by Latin letter quicker than typing the Lao letter by using 2 input method I have mentioned above , sometimes it makes mistake or misunderstanding among users, because this using Latin letter does not still obey any phonetic symbol standard but according to individual rules or habits.

It is common knowledge that people live on some nations such as China, Japan, etc… they also use Latin letter when doing work with computer applications or communicating via the internet. However, they designed only phonetic symbol standard that was used commonly and uniformly, so, using language in application on the computer become easier, faster, and cost-effective.

From the objective practice, my own initiative about designing LaoKa base on the ideal: I would design a Latin phonetic symbol that makes rules about the way to represent the Lao letters by phonetic symbol to Latin letter and named LaoKa. Basing on LaoKa phonetic symbol standard, Laoka will be designed that display Lao letters when users input Latin letters according to the rule of this phonetic symbol standard.

2.2. Advantage and disadvantage.

2.2.1. Advantage

It suits users' taste in the practice and when users input data with using Lao phonetic symbol by Latin letter so it closes to speaking language and it is easy to remember rules in Laoka phonetic symbol standard. In addition, Lao writing language does not distinguish between capital and lowercase letter so when typing Lao phonetic symbol by Latin letter, users only type right phonetic symbol. It means that only by using Latin letter keys, users can input total information but for using key combination. This makes good LaosScript and LaosUnikey's shortcomings.

2.2.2. Disadvantage

Using LaoKa compels us to learn about Latin letters, but it is no matter, because, in practice, Latin letters has been taught as a basic subject in almost of the school in Laos.

3. SOME INTRODUCTION TO LANGUAGE AND WORD STRUCTURE IN LAO AND LAOKA STANDARD.

3.1. Lao language.

Like other language on the world, Lao word is combined by different syllables, however, short and long syllable are two main syllables in Lao language. So in the Lao word, letters are divided into two main kinds to present for short syllables and long syllables. Each syllable is created from vowels or the association of vowels and consonants .

3.1.1. Consonant in Lao language.

There are two main kind of consonant: consonant and double consonant.

besides consonants are divided into uncombined and combine consonant.

- Uncombinable consonant can not combine with phonetic mark when being at the end of one word.
- Combinable consonant can combine with phonetic mark even when being at the end of word.

3.1.2. The vowel in Lao language.

There are 4 main kind of vowel:

- the vowel can not combine with consonant to become syllable.
- the vowel combine compulsorily to become syllable: this vowel can not stand alone but it must be combined with consonant to become syllable.
- the vowel can combine with consonant to become syllable.
- double vowel: this vowel can combine various one to become syllable.

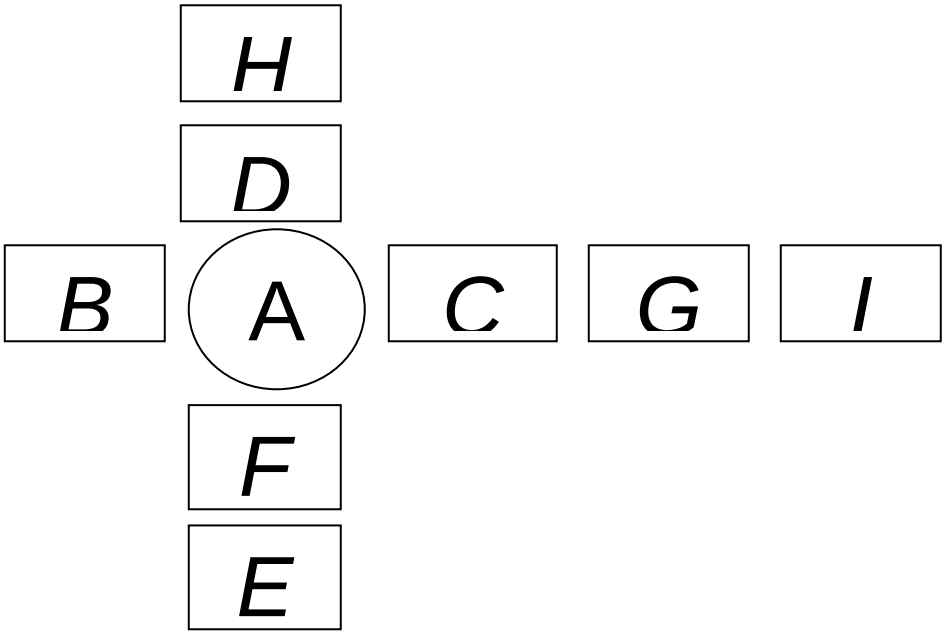
3.1.3. Sign in Lao word.

There are voiced sign and special sign:

- voiced sign: ‘ech’; ‘tho’; ‘ti’; ‘chattava’ sign.
- special sign: karan sign; repeating sign.

3.1.4. Word structure.

General structure of Laos word is below:



- A :the first consonant (rooted consonant).
- B : the vowel stands forward the first consonant.
- C : the vowel stands backward the first consonant.
- D : the vowel stands right upper of the first consonant.
- E: the vowel stands right down of the first consonant.
- F : the sign is to replace ‘for L ‘ sound element when using to combine into double consonant.
- G : sound element is used to combine with one of A/B/C/D vowel.
- H: voiced sign.
- I: special sign.

In the one Lao word, the other part depend on different word, but the fist consonant is obligated.

3.2. LaoKa standard.

Like mentioning in other part above, Laoka standard is buit by latin letter to represent for Lao language.

3.2.1. consonant and double consonant standard.

Consonant standard table:

Roll number	LaoKa	Lao language	Note
1	G	ก	*
2	KH	ข	
3	K	ค	
4	NG	ง	**
5	J	จ	
6	S	ส	
7	C	ฉ	
8	NH	ย	**
9	D	ด	*
10	T	ต	

11	THH	ฑ	
12	TH	ฒ	
13	N	ณ	**
14	B	ด	*
15	P	ต	
16	PH	ถ	
17	PHH	ท	
18	F	ธ	
19	FH	น	
20	M	บ	**
21	Y	ป	
22	L	ผ	
23	V	ฝ	**
24	H	พ	
25	HH	ฟ	

26	OH	ອ	
27	R	ຣ	

Note:

- * Uncombinable consonant.
- ** Combinable consonant.

Double consonant standard table

Roll number	LaoKa	Lao language	Note
1	HHN	ໝ=ຫນ	
2	HHM	ໝ=ຫມ	
3	HHV	ຫວ	
4	HHL	ຫລ=ຫຼ	
5	HHNG	ຫງ	
6	HHNH	ຫຍ = ຫຽ	

Note: HH can only combine with N,M,V,L,NG,NH to become double consonant.

3.2.2. Vowel standard.

Vowel stanadard table:

Roll number	LaoKa	Laolanguage	Note
1	A	xะ	1
2	AA	xา	3
3	I	xิ	3
4	II	xี	3
5	W	xึ	3
6	WW	xื	3
7	U	xุ	3
8	UU	xู	3
9	EA	เxะ	1,4
10	E	เx	3
11	EEA	แxะ	1,4
12	EE	แx	3
13	OWA	โxะ	1,4
14	OW	โx	3

15	OA	◌ະ	1,4
16	O	◌ໍ	1
17	EW	◌ິ◌	3,4
18	EWW	◌ິ◌	3,4
19	IAA	◌ັຍ (◌ັ◌)	1,4
20	IA	◌ຍ (◌◌)	3,4
21	UA	◌ົ◌	3,4
22	UAA	◌ົ◌ະ	1,4
23	AY	◌◌	1
24	AAY	◌◌	1
25	AU	◌ົ◌	1,4
26	AW	◌ໍ	1,4
27	EI	◌ິ	3,4
28	EII	◌ິ	3,4
29	AE	◌ັ	2

30	OO	x◌	2
31	Q	xJ	2

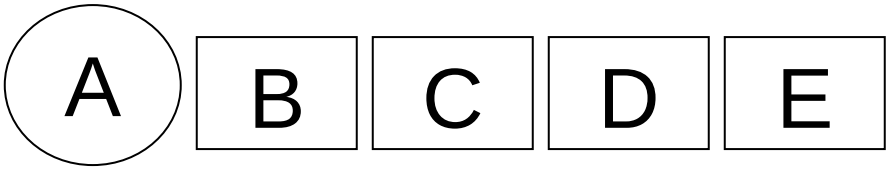
Note:

- 1 is the vowel can not combine with consonant to become syllable.
- 2 is the vowel combine compulsorily to become syllable.
- 3 is the vowel can combine with consonant to become syllable
- 4 is double vowel.

3.2.3. Sign standard.

Roll number	LaoKa	Lao language	Note
1	F	x◌	‘Ech’ sign
2	J	x◌	‘Tho’ sign
3	Z	x◌	‘Ti’ sign
4	X	x◌	‘Chattava’ sign
5	L	x◌	‘Karan’ sign
6	P	xๆ	Repeating sign

3.2.4. Word structure in LaoKa standard.



- A : the first consonant of one word.
- B: the vowel.
- C : sound element is used to combine with the vowel.
- D: voiced sign.
- E: special sign. Example: ສ້ໆ (SAANGJ), ມ້ໆ (MAANGJ), ປ່ານ (PANF*), ປ້ໆຍ (PAIJ *+).

Note:

- * is Rule II.
- *+ is Rule II+.

The position in Laoka standard is consider in the detail below:

- A: the obliged and first consonant in one word. It consists of consonant and double consonant in Laoka consonant standard table.
- Example: “ການ(GAAN)” . In this case , G is first consonant.
- B : is the vowel is used to combine with first consonant, it can combine with sound element depending on various vowels. Example: “(PAAN)”, “(SA)”. In this case ,“AA” and “A” are vowels.
- C: sound element is used to combine with one of vowel to become syllable. Example: “TAANH” . In this case , “NH” is sound element.
- D: Voiced sign that mention in LaoKa standard sign table consists of 4 sign. Example: “ PAANHJ” . In this case,’J’ is voiced sign.
- E:Special sign that is displayed in LaoKa standard sign table ,include 2 sign:Karan sign and repeating sign.

- Karan sign is only used to represent from English to Lao. Example: ແມກສ์(MEEGSL) is phonetic Max, ແບກ์ (BEEGKL) is phonetic Back. In this case ‘L’ is Karan sign.
- Repeating sign is used to repeat one word which stand right forward it. Example: ໄປໆ(PAYP) .In this case, ‘P’ is repeating sign.

3.2.5. LaoKa’s rules.

Rule I: To type one word in Lao by LaoKa’s rules.This below structure is relied on.:

Consonant + Vow l + Sound Element + Voiced Sign + Special Sign

- Example:
  - ປາ (P+AA), include: Consonant + Vowel.
  - ນ (S+AE+N), include: Consonant + Vowel + Sound Element.
  - ນັ້ນ (N+AE+N+J), include: Consonant + Vowel + Sound Element + Voiced Sign.
  - ອື່ນໆ (OH+WW+N+F+P), include: Consonant + Vowel + Sound Element + Voiced Sign + Special Sign.

It means that Rule I is used originally by ruled Laoka standard table.

Rule II: This rule is built to help us type faster and more comviniently than Rule I because when Rule II is used ,Rule I can also used through Rule II.

- The word begins with vowel it will understand by itself that it contains ‘ອ’(OH) consonant which is consider as the first consonant. Example: WWNFP =***OH***WWNFP (ອື່ນໆ), IIG = ***OH***IIG (ອີກ) , AEN=***OH***AEN (ອັນ)....

- Because ‘xะ’ (A) vowel is uncombined vowel so in this case,the ‘xะ’ (A) vowel will replace for ‘xา’ (AA) vowel to combine into syllable. Example: OH***A***NF = OH***AA***NF =ອ่ານ , TH***A***NG = TH***AA***NG =ທາງ....
- ‘x̊’ (O) vowel is similar to the case above, ‘x̊’ (O) vowel will be replace for ‘ອ’ (OH) consonant to combine into syllable. Example: MOB = MOHB = ມອບ , OHOHG = OHOG = OG=ອອກ…
- As for ‘ຫ’ (HH) consonant, it is used to combine with other consonant:N,M,NG,L,V,NH to become double consonant,but in Rule II (H) is replace for ‘ຫ’ (HH). Except for special case of V consonant , ‘ຫ’ (HH) consonant is also used as usual to avoid misunderstanding in Lao language. Example: ໄຫ້(***H***MAYJ = ***HH***MAYJ)…

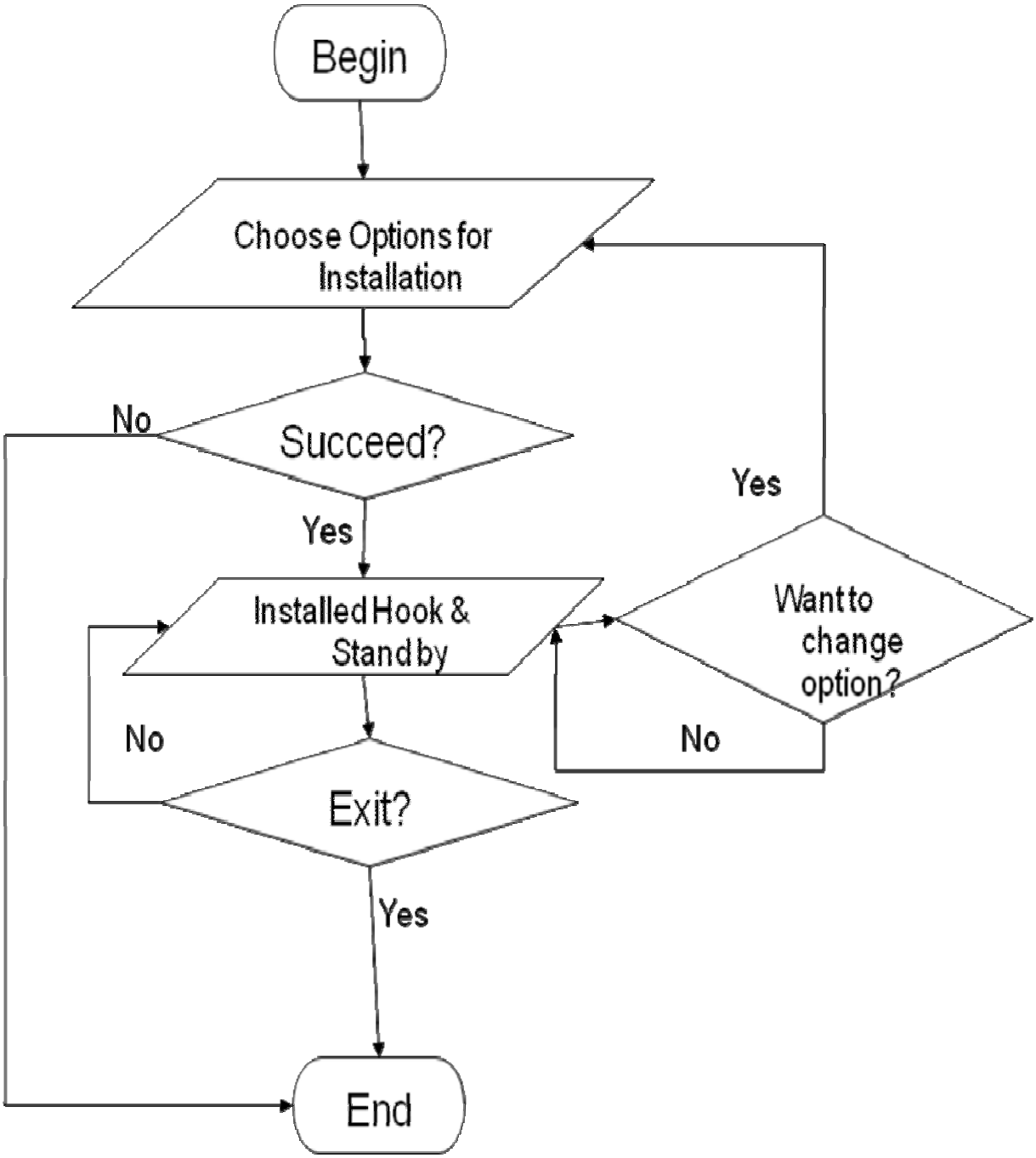
Rule II+: Beside Rule I and Rule II,some other Rule is also used to help typing faster even than Rule II.

- The Rules is represented below:
  - **-AI = ANH = AANH =xາຍ.**
  - **-OI = ONH =OHNH= xອຍ.**
  - **-OY= OWNH = ໂxຍ.**
  - **-AO = AV = AAV = xາວ.**
  - **-UI = UNH = xຸ̱ຍ.**
  - **-UY = UUNH=xູ̱ຍ.**
  - **-EO = EEV =ແxວ.**

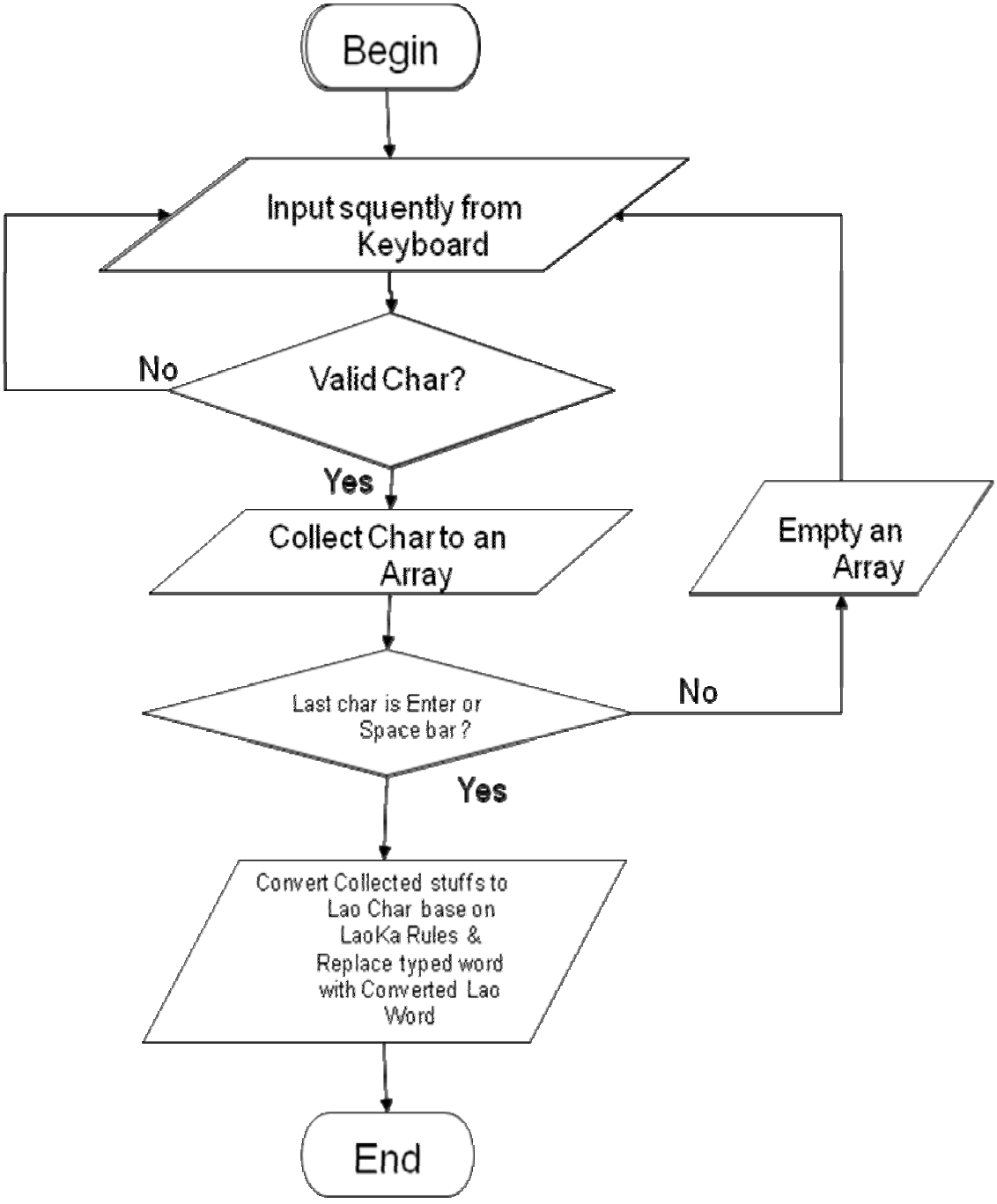
3.3. Algorithm.

3.3.1. Flow chart.

Installation/Uninstallation:



Processing.



3.3.2. Procedure of LaoKa converter.

After installing LaoKa program and starting the program to use. If user type any text and program while starting LaoKa program .At the first time user type the letters

from keyboard to editor, if Caplocks key is turned off , Lao typing function is considered to be off and in case, Caplocks key is turned on ,it means that Lao typing function is on and LaoKa program will perform a mass of following implement:

- The character which user type (A-Z. 1-9) , apart from key number of numlock part before pressing space key, is expressed on editor as typed steps. These characters is saved in array in memory to wait for exchange from Latin to Lao character. In case user has not yet press enter or space key ,if user deletes characters by using backspace key , these characters will be delete from interface of editor and array in memory automatically at the same time .If user clicks mouse , the typed characters will be kept in editor but it is deleted from array in memory.
- After typing characters and user press space key, LaoKa program will exchange from saved Latin in memory to Lao characters formally and then delete Latin character on editor and replaced by correlative Lao character .Exchanging is relied on Laoka standard, exchanging machanism of each program is different , however, all machanism obey following exchanging step:
- The program will test in saved array of character which user have just typed whether which character is voiced sign as in ruled Laoka standard. If yes, this sign will change into an array which is called as voiced sign array.
- The rest of sign will be changed into other array which is named as exchanging array), and then the program will perform ordering position of Latin sign in exchanging array in the term of right structure in LaoKa standard. Ordering position is as follow:
  - If the vowels is the kind of vowel which stand right forward of first consonant, the program will put that vowels on forward position of that first consonant. The vowels of this kind such as: AY,AAY…
  - If the last position is P or L letter , the program will self-understand this sign is special sign and is still kept in last position of exchanging array. After these characters is ordered again and marked to distinguish other L or P consonant in the same array. In

other hand, this position is consider as the sign but it’s not the consonant.

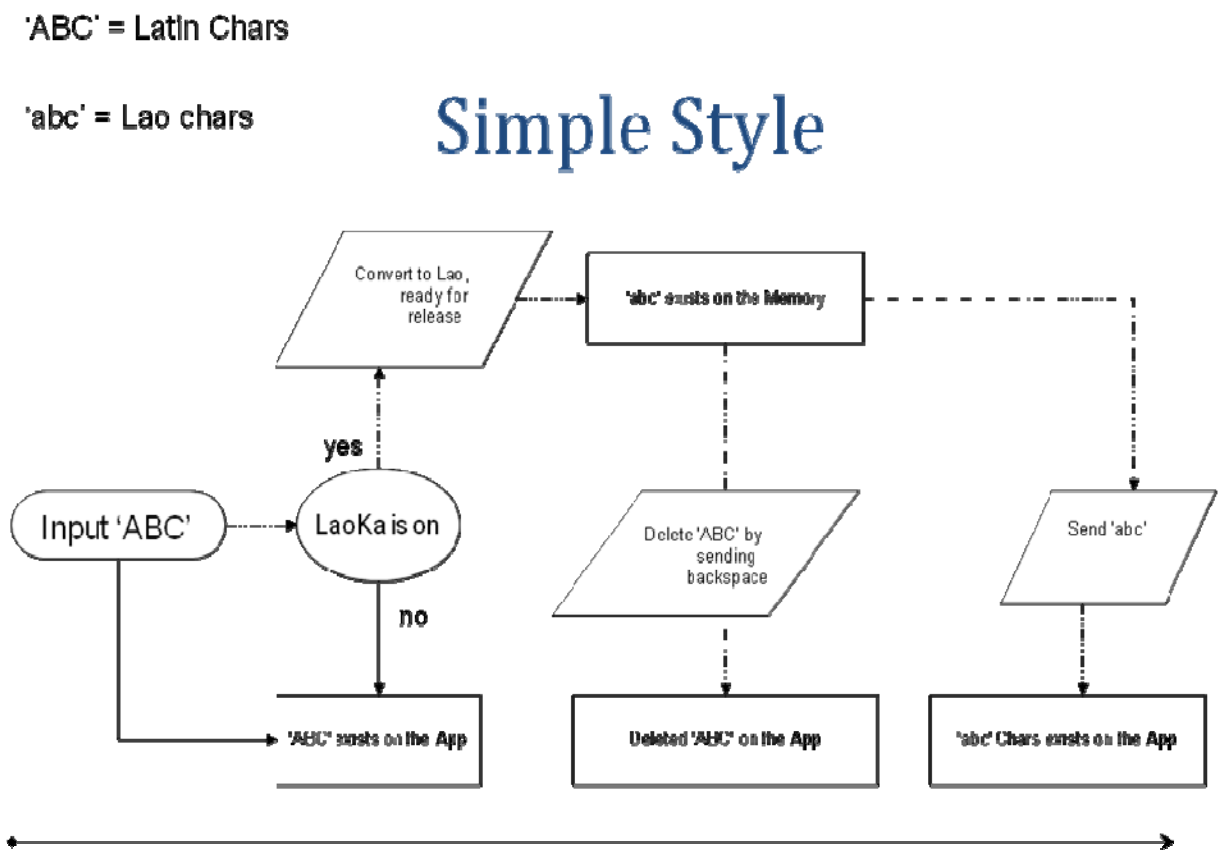
- In other case, ordering position is kept primarily.

- As for exchanging array, after the character is ordered, the program will perform to convert into Lao character as the ruled LaoKa Standard, and then the program will delete Latin characters on editor and is replaced by correlative Lao character.
- Finally, The program will reset all of variables.

3.3.3. Operating machanism of Simple Style and Unicode Style.

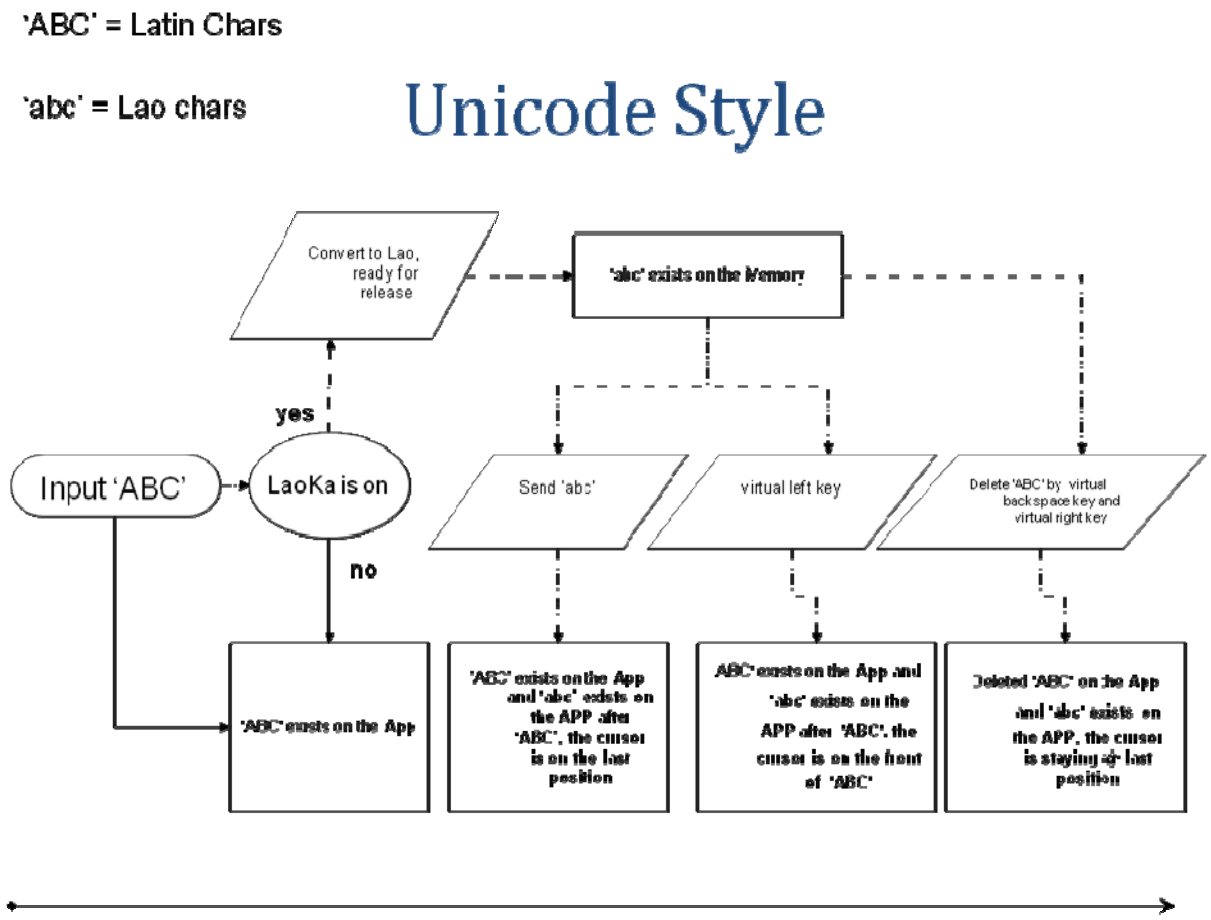
Operating machanism of Simple Style.

- When performing deleting of Latin character on editor and replace them by Lao character, the program will send each backspace command to editor to delete Latin character , and then send correlative Lao character.
- This operating machanism of Simple Style is not suitable for the program which uses Hook such as Yahoo messenger…but it suits the program such as: Msword ,notepad…
- This below flowchart will illustrate for Simple Style:



Operating machanism of Unicode Style.

- The program will send Lao character to editor and stand next position of Latin character, after that, it change cursor to in front of Latin character and then delete it.
- This machanism will suits the program which use Hooks because of using virtual key to delete Latin character.
- This below flowchart will illustrate for Unicode Style:



3.4. Lao Unicode.

3.4.1. Range: 0E80-0EFF.

This file contains an excerpt from the character code tables and list of character names for *The Unicode Standard ,version 5.1*.

This file may be changed at any time without notice to reflect errata or other updates to the Unicode Standard.

- See http://www.unicode.org/errata /for an up-to-date list of errata .
- See http://www.unicode.org/charts /for access to a complete list of the latest character code charts.
- See http://www.unicode.org/charts/PDF/Unicode-5.1/ for charts showing only the characters added in Unicode 5.1.

- See http://unicode.org/Public/5.1.0/charts/ for a complete archived file of character code charts for Unicode 5.1.

3.4.2. Disclaimer.

This charts are provided as the online reference to the character contents of the Unicode Standard ,Version 5.1 but not provide all the information needed to fully support individual scripts using the Unicode Standard. For the complete understanding of the use of the characters contained in this file, please consult to appropriated sections of The Unicode Standard ,Version 5.0 (ISBN: 0-321-48091-0) ,online at http://www.unicode.org/versions/Unicode5.0.0/ ,as well as Unicode Standard Annexes #9,#11,#14,#15.#24,#29,#31,#34,#38,#41,#42, and #44 , the others Unicode technical Reports and Standards , and the Unicode Character Database ,which are available online. See http://www.unicode.org/ucd / and http://www.unicode.org/reports/

A thorough understanding the information contained in these additional sources is required for successful implementation.

3.4.3. Fonts.

The shapes of the reference glyphs used in these code charts are not prescriptive. Considerable variation is to be expected in actual fonts. The particular fonts used in these charts were provided to the Unicode Consortium by the number of difference font designers, who own the rights to the fonts. See http://www.unicode.org/charts/fonts.html for a list.

3.4.4. Terms of Use

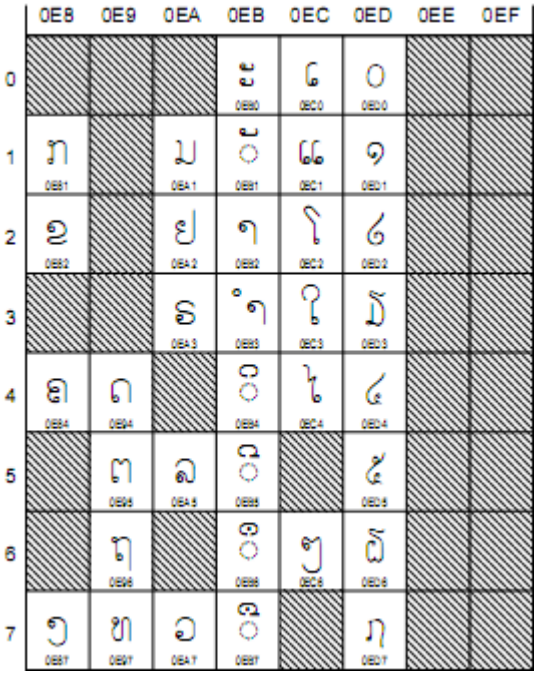
You may freely use these code charts for personal or internal business use only. You may not incorporation them either wholly or in part into any product or publication , or otherwise distribute them without express written permission from the Unicode Consortium. However, you may provide link to these charts.

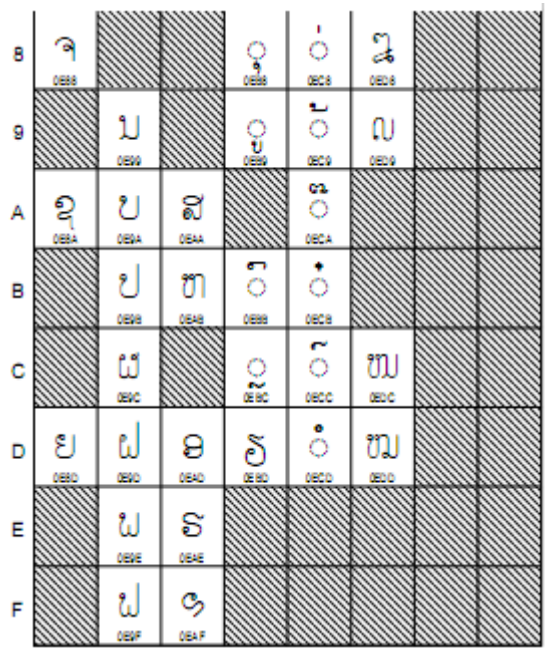
The fonts and font data used in production of these code charts may not be extracted, or used in any other way in any product or publication, without permission or license granted by the typeface owner(s).

The Unicode Consortium is not liable for errors or omissions in this file or the standard itself. Information on characters added to the Unicode Standard since the

publication of the most recent version of the Unicode Standard, as well as on characters currently being considered for addition to the Unicode Standard can be found on the Unicode website. See http://www.unicode.org/pending/pending.html and http://www.unicode.org/alloc/Pipeline.html .

0E80 Lao 0EFF





The Unicode Standard 5.1, Copyright © 1991-2008 Unicode, Inc. All rights reserved.

4. LAOKA INSTALLATION.

4.1. Hooks.

A hook is a point in the system message-handling mechanism where an application can install a subroutine to monitor the message traffic in the system and process certain types of messages before they reach the target window procedure.

4.1.1. About Hooks.

Hooks tend to slow down the system because they increase the amount of processing the system must perform for each message. You should install a hook only when necessary, and remove it as soon as possible.

This section discusses the following:

- Hook Chains.
  - The system supports many different types of hooks; each type provides access to a different aspect of its message-handling mechanism. For example, an application can use the WH_MOUSE Hook to monitor the message traffic for mouse messages.

- The system maintains a separate hook chain for each type of hook. A *hook chain* is a list of pointers to special, application-defined callback functions called *hook procedures*. When a message occurs that is associated with a particular type of hook, the system passes the message to each hook procedure referenced in the hook chain, one after the other. The action a hook procedure can take depends on the type of hook involved. The hook procedures for some types of hooks can only monitor messages; others can modify messages or stop their progress through the chain, preventing them from reaching the next hook procedure or the destination window.

o Hook Procedures.

- To take advantage of a particular type of hook, the developer provides a hook procedure and uses the SetWindowsHookEx function to install it into the chain associated with the hook. A hook procedure must have the following syntax:

```
LRESULT CALLBACK HookProc(
  int nCode,
  WPARAM wParam,
  LPARAM lParam
);
```

- *HookProc* is a placeholder for an application-defined name.
- The *nCode* parameter is a hook code that the hook procedure uses to determine the action to perform. The value of the hook code depends on the type of the hook; each type has its own characteristic set of hook codes. The values of the *wParam* and

lParam parameters depend on the hook code, but they typically contain information about a message that was sent or posted.

- The **SetWindowsHookEx** function always installs a hook procedure at the beginning of a hook chain. When an event occurs that is monitored by a particular type of hook, the system calls the procedure at the beginning of the hook chain associated with the hook. Each hook procedure in the chain determines whether to pass the event to the next procedure. A hook procedure passes an event to the next procedure by calling the CallNextHookEx function.
- Note that the hook procedures for some types of hooks can only monitor messages. the system passes messages to each hook procedure, regardless of whether a particular procedure calls **CallNextHookEx**.
- A *global hook* monitors messages for all threads in the same desktop as the calling thread. A *thread-specific hook* monitors messages for only an individual thread. A global hook procedure can be called in the context of any application in the same desktop as the calling thread, so the procedure must be in a separate DLL module. A thread-specific hook procedure is called only in the context of the associated thread. If an application installs a hook procedure for one of its own threads, the hook procedure can be in either the same module as the rest of the application's code or in a DLL. If the application installs a hook procedure for a thread of a different application, the procedure must be in a DLL. For information, see Dynamic-Link Libraries.
- **Note** You should use global hooks only for debugging purposes; otherwise, you should avoid them. Global hooks hurt system performance and cause conflicts with other applications that implement the same type of global hook.

o Hook Types.

- Each type of hook enables an application to monitor a different aspect of the system's message-handling mechanism. The following sections describe the available hooks.
- WH_CALLWNDPROC and WH_CALLWNDPROCRET Hooks
- WH_CBT Hook
- WH_DEBUG Hook
- WH_FOREGROUNDIDLE Hook
- WH_GETMESSAGE Hook
- WH_JOURNALPLAYBACK Hook
- WH_JOURNALRECORD Hook
- WH_KEYBOARD_LL Hook
- WH_KEYBOARD Hook
- WH_MOUSE_LL Hook
- WH_MOUSE Hook
- WH_MSGFILTER and WH_SYSMSGFILTER Hooks
- WH_SHELL Hook
- WH_CALLWNDPROC and WH_CALLWNDPROCRET Hooks
- The WH_CALLWNDPROC and WH_CALLWNDPROCRET hooks enable you to monitor messages sent to window procedures. The system calls a WH_CALLWNDPROC hook procedure before passing the message to the receiving window procedure, and calls the WH_CALLWNDPROCRET hook procedure after the window procedure has processed the message.
- The WH_CALLWNDPROCRET hook passes a pointer to a CWPRETSTRUCT structure to the hook procedure. The structure contains the return value from the window procedure that processed the message, as well as the message parameters

associated with the message. Subclassing the window does not work for messages set between processes.

- For more information, see the CallWndProc and CallWndRetProc functions.

- WH_CBT Hook.
  - The system calls a WH_CBT hook procedure before activating, creating, destroying, minimizing, maximizing, moving, or sizing a window; before completing a system command; before removing a mouse or keyboard event from the system message queue; before setting the input focus; or before synchronizing with the system message queue. The value the hook procedure returns determines whether the system allows or prevents one of these operations. The WH_CBT hook is intended primarily for computer-based training (CBT) applications.
  - For more information, see the CBTProc function.
  - For information, see WinEvents.
- WH_DEBUG Hook
  - The system calls a WH_DEBUG hook procedure before calling hook procedures associated with any other hook in the system. You can use this hook to determine whether to allow the system to call hook procedures associated with other types of hooks.
  - For more information, see the DebugProc function.
- WH_FOREGROUNDIDLE Hook
  - The WH_FOREGROUNDIDLE hook enables you to perform low priority tasks during times when its foreground thread is idle. The system calls a WH_FOREGROUNDIDLE hook procedure when the application's foreground thread is about to become idle.
  - For more information, see the ForegroundIdleProc function.
- WH_GETMESSAGE Hook.

- The WH_GETMESSAGE hook enables an application to monitor messages about to be returned by the GetMessage or PeekMessage function. You can use the WH_GETMESSAGE hook to monitor mouse and keyboard input and other messages posted to the message queue.
- For more information, see the GetMsgProc function.

- WH_JOURNALPLAYBACK Hook
  - The WH_JOURNALPLAYBACK hook enables an application to insert messages into the system message queue. You can use this hook to play back a series of mouse and keyboard events recorded earlier by using the WH_JOURNALRECORD Hook. Regular mouse and keyboard input is disabled as long as a WH_JOURNALPLAYBACK hook is installed. A WH_JOURNALPLAYBACK hook is a global hook — it cannot be used as a thread-specific hook.
  - The WH_JOURNALPLAYBACK hook returns a time-out value. This value tells the system how many milliseconds to wait before processing the current message from the playback hook. This enables the hook to control the timing of the events it plays back.
  - For more information, see the JournalPlaybackProc function.
- WH_JOURNALRECORD Hook
  - The WH_JOURNALRECORD hook enables you to monitor and record input events. Typically, you use this hook to record a sequence of mouse and keyboard events to play back later by using the WH_JOURNALPLAYBACK Hook. The WH_JOURNALRECORD hook is a global hook — it cannot be used as a thread-specific hook.
  - For more information, see the JournalRecordProc function.
- WH_KEYBOARD_LL Hook

- The WH_KEYBOARD_LL hook enables you to monitor keyboard input events about to be posted in a thread input queue.
- For more information, see the LowLevelKeyboardProc function.

- WH_KEYBOARD Hook
  - The WH_KEYBOARD hook enables an application to monitor message traffic for WM_KEYDOWN and WM_KEYUP messages about to be returned by the **GetMessage** or **PeekMessage** function. You can use the WH_KEYBOARD hook to monitor keyboard input posted to a message queue.
  - For more information, see the KeyboardProc function.
- WH_MOUSE_LL Hook
  - The WH_MOUSE_LL hook enables you to monitor mouse input events about to be posted in a thread input queue.
  - For more information, see the LowLevelMouseProc function.
- WH_MOUSE Hook
  - The WH_MOUSE hook enables you to monitor mouse messages about to be returned by the **GetMessage** or **PeekMessage** function. You can use the WH_MOUSE hook to monitor mouse input posted to a message queue.
  - For more information, see the MouseProc function.
- WH_MSGFILTER and WH_SYSMSGFILTER Hooks
  - The WH_MSGFILTER and WH_SYSMSGFILTER hooks enable you to monitor messages about to be processed by a menu, scroll bar, message box, or dialog box, and to detect when a different window is about to be activated as a result of the user's pressing the ALT+TAB or ALT+ESC key combination. The WH_MSGFILTER hook can only monitor messages passed to a menu, scroll bar, message box, or dialog box created by the application that installed the hook procedure. The

WH_SYSMSGFILTER hook monitors such messages for all applications.

- The WH_MSGFILTER and WH_SYSMSGFILTER hooks enable you to perform message filtering during modal loops that is equivalent to the filtering done in the main message loop. For example, an application often examines a new message in the main loop between the time it retrieves the message from the queue and the time it dispatches the message, performing special processing as appropriate. However, during a modal loop, the system retrieves and dispatches messages without allowing an application the chance to filter the messages in its main message loop. If an application installs a WH_MSGFILTER or WH_SYSMSGFILTER hook procedure, the system calls the procedure during the modal loop.
- An application can call the WH_MSGFILTER hook directly by calling the CallMsgFilter function. By using this function, the application can use the same code to filter messages during modal loops as it uses in the main message loop. To do so, encapsulate the filtering operations in a WH_MSGFILTER hook procedure and call **CallMsgFilter** between the calls to the **GetMessage** and DispatchMessage functions.

```
while (GetMessage(&msg, (HWND) NULL, 0, 0))
{
  if (!CallMsgFilter(&qmsg, 0))
     DispatchMessage(&qmsg);
}
```

- The last argument of **CallMsgFilter** is simply passed to the hook procedure; you can enter any value. The hook procedure, by defining a constant such as MSGF_MAINLOOP, can use this value to determine where the procedure was called from.
- For more information, see the MessageProc and SysMsgProc functions.

- WH_SHELL Hook
  - A shell application can use the WH_SHELL hook to receive important notifications. The system calls a WH_SHELL hook procedure when the shell application is about to be activated and when a top-level window is created or destroyed.
  - Note that custom shell applications do not receive WH_SHELL messages. Therefore, any application that registers itself as the default shell must call the SystemParametersInfo function before it (or any other application) can receive WH_SHELL messages. This function must be called with SPI_SETMINIMIZEDMETRICS and a MINIMIZEDMETRICS structure. Set the **iArrange** member of this structure to ARW_HIDE.
  - For more information, see the ShellProc function.

4.1.2. Functions.

CallMsgFilter.

- The CallMsgFilter function passes the specified message and hook code to the hook procedures associated with the WH_SYSMSGFILTER and WH_MSGFILTER hooks. A **WH_SYSMSGFILTER** or **WH_MSGFILTER** hook procedure is an application-defined callback function that examines and, optionally, modifies messages for a dialog box, message box, menu, or scroll bar.

CallNextHookEx.

- The CallNextHookEx function passes the hook information to the next hook procedure in the current hook chain. A hook procedure can call this function either before or after processing the hook information.

CallWndProc.

- The CallWndProc hook procedure is an application-defined or library-defined callback function used with the SetWindowsHookEx function.
- **Windows 95/98/Me, Windows NT 3.51:** The system calls this function whenever the thread calls the SendMessage function. The WH_CALLWNDPROC hook is called in the context of the thread that calls **SendMessage**, not the thread that receives the message.
- **Windows NT 4.0 and later:** The system calls this function before calling the window procedure to process a message sent to the thread.
- The **HOOKPROC** type defines a pointer to this callback function. **CallWndProc** is a placeholder for the application-defined or library-defined function name.

CallWndRetProc.

- The CallWndRetProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function after the **SendMessage** function is called. The hook procedure can examine the message; it cannot modify it.
- The **HOOKPROC** type defines a pointer to this callback function. **CallWndRetProc** is a placeholder for the application-defined or library-defined function name.

CBTProc.

- The CBTProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function before activating, creating, destroying, minimizing, maximizing, moving, or sizing a window; before completing a system command; before removing a mouse or keyboard event from the

system message queue; before setting the keyboard focus; or before synchronizing with the system message queue. A computer-based training (CBT) application uses this hook procedure to receive useful notifications from the system.

- The **HOOKPROC** type defines a pointer to this callback function. **CBTProc** is a placeholder for the application-defined or library-defined function name.

DebugProc.

- The DebugProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function before calling the hook procedures associated with any type of hook. The system passes information about the hook to be called to the **DebugProc** hook procedure, which examines the information and determines whether to allow the hook to be called.
- The **HOOKPROC** type defines a pointer to this callback function. **DebugProc** is a placeholder for the application-defined or library-defined function name.

ForegroundIdleProc.

- The ForegroundIdleProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function whenever the foreground thread is about to become idle.
- The **HOOKPROC** type defines a pointer to this callback function. **ForegroundIdleProc** is a placeholder for the application-defined or library-defined function name.

GetMsgProc.

- The GetMsgProc function is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function whenever the GetMessage or PeekMessage function has retrieved a message from an application message queue.

Before returning the retrieved message to the caller, the system passes the message to the hook procedure.

- The **HOOKPROC** type defines a pointer to this callback function. **GetMsgProc** is a placeholder for the application-defined or library-defined function name.

JournalPlaybackProc.

- The JournalPlaybackProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. Typically, an application uses this function to play back a series of mouse and keyboard messages recorded previously by the JournalRecordProc hook procedure. As long as a **JournalPlaybackProc** hook procedure is installed, regular mouse and keyboard input is disabled.
- The **HOOKPROC** type defines a pointer to this callback function. **JournalPlaybackProc** is a placeholder for the application-defined or library-defined function name.

JournalRecordProc.

- The **JournalRecordProc** hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The function records messages the system removes from the system message queue. Later, an application can use a **JournalPlaybackProc** hook procedure to play back the messages.
- The **HOOKPROC** type defines a pointer to this callback function. **JournalRecordProc** is a placeholder for the application-defined or library-defined function name.

KeyboardProc.

- The KeyboardProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function whenever an application calls the

GetMessage or **PeekMessage** function and there is a keyboard message (WM_KEYUP or WM_KEYDOWN) to be processed.

- The **HOOKPROC** type defines a pointer to this callback function. **KeyboardProc** is a placeholder for the application-defined or library-defined function name.

LowLevelKeyboardProc.

- The LowLevelKeyboardProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function every time a new keyboard input event is about to be posted into a thread input queue. The keyboard input can come from the local keyboard driver or from calls to the keybd_event function. If the input comes from a call to **keybd_event**, the input was "injected". However, the WH_KEYBOARD_LL hook is not injected into another process. Instead, the context switches back to the process that installed the hook and it is called in its original context. Then the context switches back to the application that generated the event.
- The **HOOKPROC** type defines a pointer to this callback function. **LowLevelKeyboardProc** is a placeholder for the application-defined or library-defined function name.

LowLevelMouseProc.

- The LowLevelMouseProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system call this function every time a new mouse input event is about to be posted into a thread input queue. The mouse input can come from the local mouse driver or from calls to the mouse_event function. If the input comes from a call to **mouse_event**, the input was "injected". However, the WH_MOUSE_LL hook is not injected into another process. Instead, the context switches back to the process that installed the hook and it is called in its original context. Then the context switches back to the application that generated the event.

- The **HOOKPROC** type defines a pointer to this callback function. **LowLevelMouseProc** is a placeholder for the application-defined or library-defined function name.

MessageProc.

- The MessageProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function after an input event occurs in a dialog box, message box, menu, or scroll bar, but before the message generated by the input event is processed. The hook procedure can monitor messages for a dialog box, message box, menu, or scroll bar created by a particular application or all applications.
- The **HOOKPROC** type defines a pointer to this callback function. **MessageProc** is a placeholder for the application-defined or library-defined function name.

MouseProc.

- The MouseProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function whenever an application calls the **GetMessage** or **PeekMessage** function and there is a mouse message to be processed.
- The **HOOKPROC** type defines a pointer to this callback function. **MouseProc** is a placeholder for the application-defined or library-defined function name.

SetWindowsHookEx.

- The **SetWindowsHookEx** function installs an application-defined hook procedure into a hook chain. You would install a hook procedure to monitor the system for certain types of events. These events are associated either with a specific thread or with all threads in the same desktop as the calling thread.

ShellProc.

- The ShellProc hook procedure is an application-defined or library-defined callback function used with the **SetWindowsHookEx** function. The function receives notifications of Shell events from the system.
- The **HOOKPROC** type defines a pointer to this callback function. **ShellProc** is a placeholder for the application-defined or library-defined function name.

SysMsgProc.

- The SysMsgProc hook procedure is a library-defined callback function used with the **SetWindowsHookEx** function. The system calls this function after an input event occurs in a dialog box, message box, menu, or scroll bar, but before the message generated by the input event is processed. The function can monitor messages for any dialog box, message box, menu, or scroll bar in the system.
- The **HOOKPROC** type defines a pointer to this callback function. **SysMsgProc** is a placeholder for the application-defined or library-defined function name.

UnhookWindowsHookEx.

- The UnhookWindowsHookEx function removes a hook procedure installed in a hook chain by the **SetWindowsHookEx** function.

4.1.3. Notifications.

WM_CANCELJOURNAL.

- The WM_CANCELJOURNAL message is posted to an application when a user cancels the application's journaling activities. The message is posted with a NULL window handle.

WM_QUEUESYNC.

- The WM_QUEUESYNC message is sent by a CBT application to separate user-input messages from other messages sent through the WH_JOURNALPLAYBACK Hook procedure.

4.1.4. Structures.

CBT_CREATEWND.

- The CBT_CREATEWND structure contains information passed to a WH_CBT hook procedure, **CBTProc**, before a window is created.

CBTACTIVATESTRUCT.

- The CBTACTIVATESTRUCT structure contains information passed to a WH_CBT hook procedure, **CBTProc**, before a window is activated.

CWPRETSTRUCT.

- The CWPRETSTRUCT structure defines the message parameters passed to a WH_CALLWNDPROCRET hook procedure, **CallWndRetProc**.

CWPSTRUCT.

- The CWPSTRUCT structure defines the message parameters passed to a WH_CALLWNDPROC hook procedure, **CallWndProc**.

DEBUGHOOKINFO.

- The DEBUGHOOKINFO structure contains debugging information passed to a WH_DEBUG hook procedure, **DebugProc**.

EVENTMSG.

- The EVENTMSG structure contains information about a hardware message sent to the system message queue. This structure is used to store message information for the **JournalPlaybackProc** callback function.

KBDLLHOOKSTRUCT.

- The KBDLLHOOKSTRUCT structure contains information about a low-level keyboard input event.

MOUSEHOOKSTRUCT.

- The MOUSEHOOKSTRUCT structure contains information about a mouse event passed to a WH_MOUSE hook procedure, **MouseProc**.

MOUSEHOOKSTRUCTEX.

- The MOUSEHOOKSTRUCTEX structure contains information about a mouse event passed to a WH_MOUSE hook procedure, **MouseProc**.
- This is an extension of the **MOUSEHOOKSTRUCT** structure that includes information about wheel movement or the use of the X button.

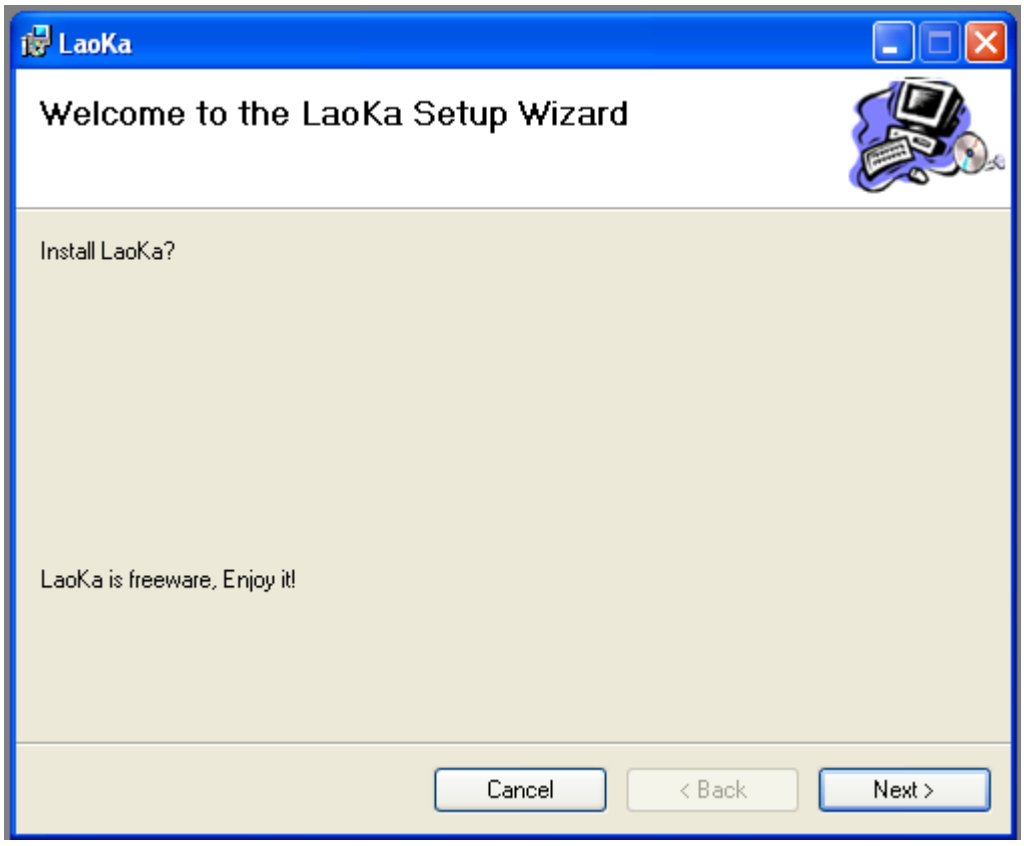
MSLLHOOKSTRUCT.

- The MSLLHOOKSTRUCT structure contains information about a low-level keyboard input event.

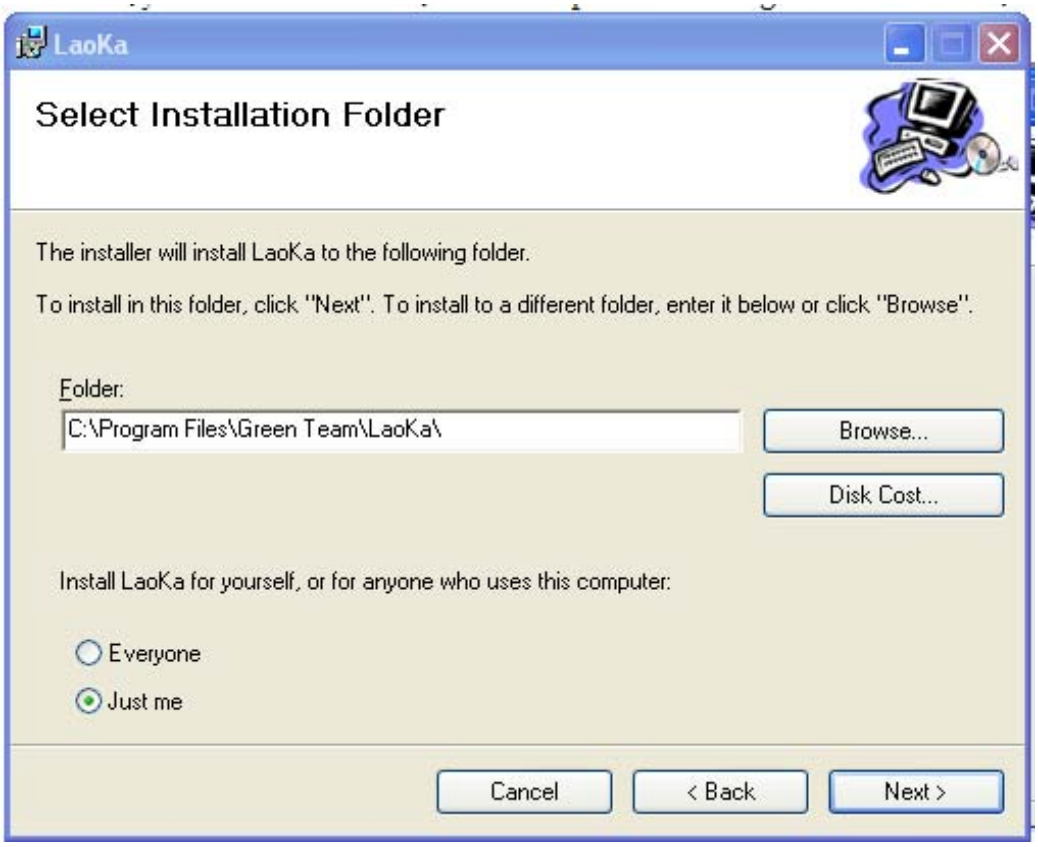
4.2. User guide for LaoKa

4.2.1. Installation :

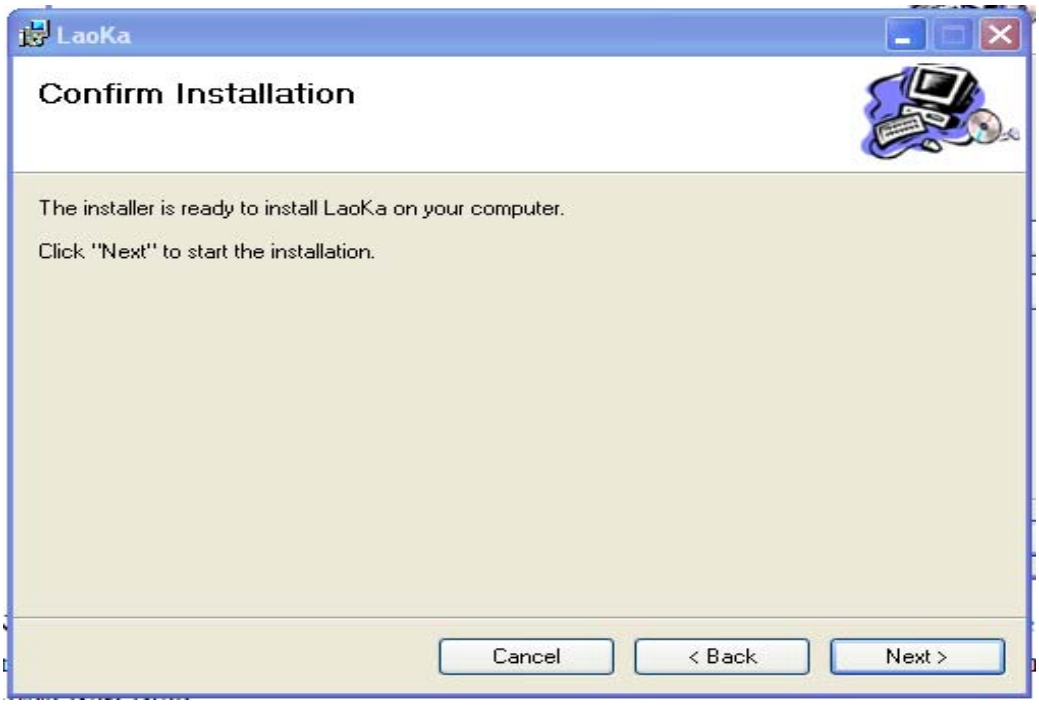
To setting Laoka program on your computer, practice by following steps



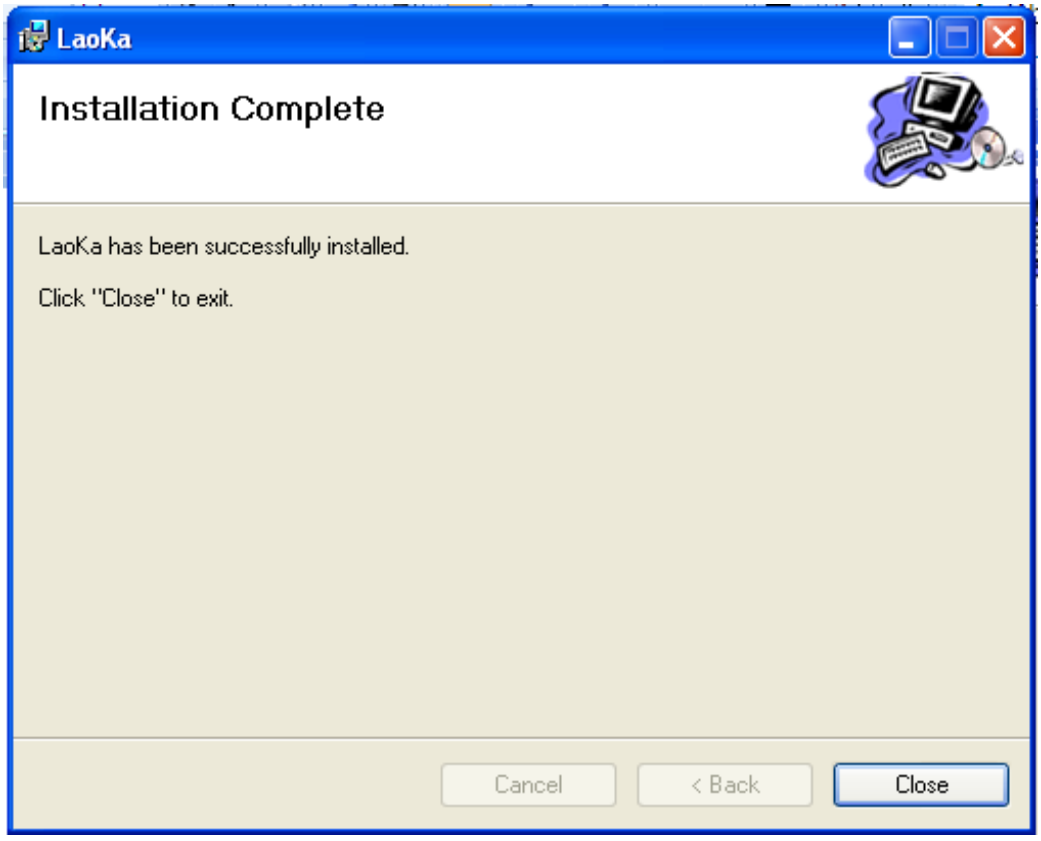
Click ‘next’button to change next screen



If you only installing for present user, please chose ‘just me’ button. If you installing for all user on the computer , please click ‘everyone’ button. Changing to next step by click ‘next’ button.



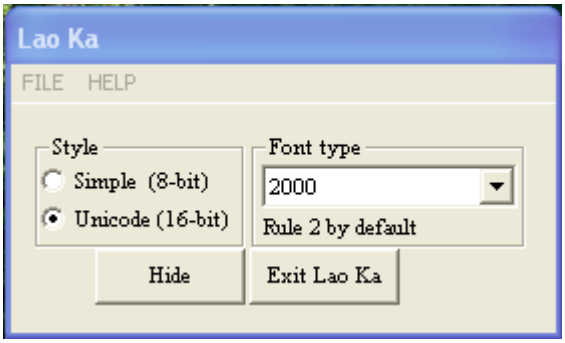
Click the mouse button on ‘next’ to install progam. Installing process, this program will show on computer screen:



Click ‘close’ button to complete setting.

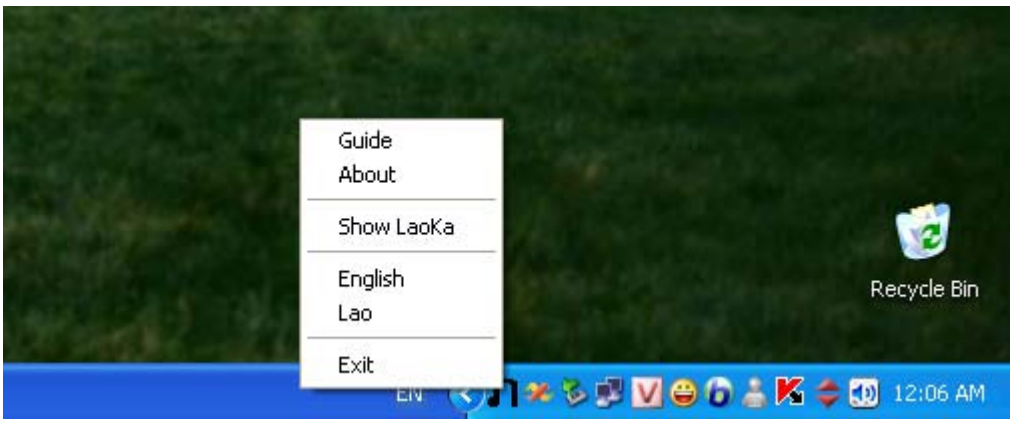
4.2.2. Use guidance.

After installing Laoka program, to use the progam, user needs to start the program . the program will display on interface (as the below image) and allows user to performfollowing action:



Note: Space key is only condition to finish one word.

- To exit the program , please choose File/Exit or click Exit LaoKa button.
- To learn information about the author of the program, please Help/About Us.
- To find out about Use Guide, please choose Help/ Use Guide
- Use Rule 2 by default
- To type Lao language on editors such as Msword , Notepad……..but having to use font saysetha2000, please choose Style is simple and Fonttype is 2000.
- To type Lao language on editors such as Msword, Notepad….but having to use unicode font such as: saysetha unicode,lao unicode…Please choose style is simple and fonttype is unicode.
- To type Lao language on chat application such as: yahoo messenger but having to use font Saysetha2000, Alice2000…Please choose style is unicode and fonttype is 2000.
- To type Lao language on chat application such as: MSN messenger and web browsers such as: IE, Firefox…but having to use unicode font such as: Saysettha unicode , Alice unicode… Please choose style is unicode and fonttype is unicode.
- When pressing 'hide' button ,an application is hidden and symbol of application is displayed on system tray .And if you right click on symbol the program will display menu which allow you choose as below image:



- To display the program on screen as the beginning time, Please choose LaoKa menu.
- To exit the program, please choose Exit menu.
- To learn about user guide, please choose Guide menu.
- To type Lao language, please choose Lao.
- To type English language, please choose English.
- To exchange from typing to not typing Lao language ,user can also use short key by pressing Caplocks key or click LaoKa symbol on system tray(at that time present language will be correlative to displayed image on symbol).

4.2.3 . Why LaoKa have to support 2000 Standard?

It's common knowledge that, in practical condition, not only any application also support typing Lao language by using Unicode Standard. Moreover, most of applications also support dramatically to type Lao language by using 2000 Standard which base on popular favoured font in Laos such as: saysettha2000,alice 2000…Therefore, LaoKa input method have just created Fonttype 2000 function to support the users to type Lao language by using 2000 Standard.

CONCLUSION

REFERENCE DOCUMENTS

[1]. About Hooks (MSDN Library – Microsoft Visual Studio 2005).
[2]. Chris Maunder, Homepage: http://www.codeproject.com/KB/shell/systemtray.aspx .
[3]. H.Joseph,Homepage:http://www.codeproject.com/KB/DLL/keyboardhook.aspx .
[4]. Homepage: http://www.unicode.org .
[5]. Hooks (MSDN Library – Microsoft Visual Studio 2005).
[6]. Jeff Prosise – Microsoft Press (1999), *Programming Windows with MFC 2nd Edition.*
[7]. Using Hooks (MSDN Library – Microsoft Visual Studio 2005).
[8]. ວິທະຍາໄລວິທະຍາສາດພື້ນຖານ(2001), *ໄວຍາກອນພາສາລາວ* (Lao Grammar).