Research Report on PDA Localization

Rajiv Bd. Jonchhay, Prajol Shrestha, Bal Krishna Bal
Kathmandu University, Madan Puraskar Pustakalaya, Nepal
john.rajiv@gmail.com, prajol@mpp.org.np, bal@mpp.org.np

Abstract

This document is a research report on the Personal Digital Assistant (PDA) localization conducted by Madan Puraskar Pustakalaya in collaboration with Kathmandu University. The whole localization process had been divided into two phases, research and development. This document provides an overview of the activities conducted in the two phase thus giving a broader picture of the PDA Localization process.

1. Introduction

The localization of handheld devices is appealing and at the same time challenging as handheld devices do not necessarily comply to the resources used by computing devices used at home or office. Development of any application for handheld devices should be guided by several factors like the reduced screen size, memory and other specific factors. Madan Puraskar Pustakalaya in collaboration with Kathmandu University conducted the PDA Localization under the PAN Localization Project. In the sections that follow, the complete process of PDA Localization is documented.

2. Research on the existing PDA Technologies

The very first task that was undertaken in PDA Localization was a research on the existing PDA Technologies. Under this topic, the basic information on PDA devices and the operating environment available in the PDA was researched. Given below are the findings of the study.

3. What is a PDA and what are its functions?

By Personal Digital Assistant (PDA) , we understand a device that can do a lot of things that a computer can do, but it is in a condensed version. All PDAs have some form of personal information management (PIM) software which is responsible for handling the organization of the following tasks [1] :

- Storing contact information
- Develop to-do lists
- Make notes
- Tracking appointments
- Reminding appointments
- Performing calculations

4. Running Application Software in the PDA

PDAs have the capability of running specialized software applications. These can be programmed by the programmer and according to the compatibility of the PDA devices they can be loaded in the respective devices [2]. Some of the most common applications are Office like applications (DataViz Documents to Go), multimedia (palmOne Media), e-mail (VersaMail), webbrowsing software etc. Others include gaming software, billing and timing software [1]. The application programs and database objects are of the format (extensions) ".prc" and ".pdb" respectively.

5. Synchronization with PCS

Owing to the fact that PDA's are designed to complement PC, the information in both of these devices should be up-to-date and hence synchronization of information should take place. For this purpose, a synchronization software on the PDA works in co-ordination with the companion software installed on the PC [1] .

6. Other existing PDA functions

In most PDAs, today, wireless and multimedia functions of some type are incorporated. Some of these functions available on most devices are as follows [3] :

- Short-range wireless connectivity through Infrared (IR) or Bluetooth technology .

- Internet and corporate network connectivity using Wi-Fi and wireless access points.

- Support for Wireless WAN (Wide Area Networks)
- A memory card slot accepting flash media such as CompactFlash, MultiMediaCard etc.
- Audio support for MP3 files and audio devices.
- A built-in digital camera
- Integrated security features like a biometric fingerprint reader.
- In-built GPS capabilities.

7. Existing operating system environment for PDA

Both proprietary and open source operating system environments exist for PDA.

8. Proprietary operating system environments

In the proprietary world of operating system environment for PDA, the Palm OS and Windows Pocket PC have been chief sellers in the world. Below in Table 1, we try to list down some of the available features of the Palm OS and Windows Pocket PC.

Table 1. PDA Devices with proprietary software

Models with Pocket PC	Models with Palm OS
HP iPaq H4350	Palm Tungsten E
Toshiba Pocket PC e750	Sony Clie
Toshiba Pocket PC e335	Palm m125
HP iPaq H5550	Garmin iQue 3600
Toshiba Pocket PC e350	
Dell Axim X5	

9. Open Source operating system environments

9.1. Qtopia:

Qtopia PDA Edition (Qtopia PDA) is the de facto standard for Linux-based PDAs. It provides a stable software platform for advanced mobile computing devices from the new generation. As evident, Qtopia runs in Linux and delivers cutting - edge functionality [11] .

9.1.1. **Qtopia features** [12] :

- Windowing system

- Synchronization framework

- Development environment

- Localization support

- Games and multimedia

- PIM applications

- Input methods

- Personalization options

- Productivity applications

- Internet applications

- Java integration

- Wireless support

9.1.2. Technical Features of Qtopia PDA [12] **:**

9.1.2.1. Source Code:

The customers of all Qtopia PDA are liable to receive the complete source code. Hence, he or she may create custom applications that integrate with the existing applications, and make compilations for using on the certain processor.

9.1.2.2. Core Platform

The base platform provides a robust computing environment aided with many useful behind-the-scene features for the end-users in terms of integration of the systems and modules. The manufacturers also may easily add extra functionality. Some of these features include input methods, adjustable screen size and orientation, plug-in manager, application installation, and wireless support.

9.1.2.3. Internationalization

As Qtopia PDA uses Unicode internally, localization can be efficiently conducted for different markets. In addition to this, the layout engine

automatically adjusts the size of the graphic user interface elements like the buttons and labels, incase, the size of the text on these elements are longer than the original text of the source language.

9.1.2.4. Customization:

Qtopis PDA has a powerful theming engine. This allows manufacturers to create custom branded user interfaces and application launchers.

9.1.3. Major PDA Models implementing Qtopia

Among Linux based PDAs, the Zaurus is the most popular and the most impressive. It costs less than Palms, however, has a lot of potential [13] .

9.2. Sharp Zaurus SL-5600:

The Sharp Zaurus SL-5600 Personal Mobile Tool is the most recently released in the 5000 series. Sharp has undergone many hardware and software improvements in the 5600 [14, 15].

For people wanting to achieve some more, it is also possible to create a customized ROM image, develop new applications, port already existing Linux applications, use the available tools to connect to a company database in the network wirelessly and administer remotely. From this perspective, the Zaurus 5600 is more than just a PDA, rather than a tiny Linux computer [15] .

9.3. Sharp Zaurus SL-6000

The new version, SL6000, is a breakthrough for the Zaurus. Among some of its features are the famous C7xx crystal-clear screen, enablement of both Wifi and Bluetooth etc. It also has a built-in, foldable keyboard very much like the old SL5xxx series. The most important features are portability, standardization, sustainability, and flexibility [16].

9.4. Kaii PDA:

The Kaii PDA (personal digital assistant) has been released by Informart, a Bangalore-based company. The hardware design of the product comes from India and the software comes from US-based Lineo. *Kaii* is based on Lineo's Embedix, an embedded Linux operating system [17].

Kaii is 'double byte enabled' so that it can support any language in the world. In addition to this, various devices such as printers, keyboards, external hard disk drives may be plugged into Kaii PDA [18].

Nevertheless, research is still ongoing in terms of the compatibility , flexibility and the GUI desktop issues of Kaii PDA . It is still unclear whether it uses Qtopis and whether it has full localization support or not.

10. Choice of the PDA model and the operating system environment

Owing to the following factors, we have opted to go for Sharp Zaurus model and Qtopia operating system environment for the purpose of PDA Localization:

1. From our study, it was found out that non-proprietary operating environment Qtopia comes up in third place in terms of wide usage after the proprietary operating environments, Palm OS and Pocket PC.
2. Among various Linux based PDA models, the Zaurus is the most popular and impressive PDA [13].
3. Resource bundle and other essentials required for localization are not readily available for proprietary operating environments.

11. Technical specifications of Sharp Zaurus SL-5500

11.1. Hardware :

Sharp Zaurus SL-5500 has the following hardware specifications [19] :

CPU – 206 Mhz Intel SA-1110 Strong ARM systems -on-chip processor;
Memory – 64 MB DRAM

Display – 3.5 inch 240 x 320 - dot pixel ("quarter VGA") reflective TFT 65,536 color LCD with touch panel support
Keyboard – front lighted QWERTY keyboard with a slide cover

Dual card slots --
– Compact Flash Type II for memory or other peripheral expansion;
– Secure Digital (SD) card slot for the secure memory storage or other peripheral expansion;

I/O Ports --
 -IrDA 1.2
 -Serial Port (via cradle)
 -USB (via cradle)
 -Stereo headset jack; includes audio input (mono)

PowerSource --
 -Lithium ion battery (950 mAH)
 -Battery life: 10 hr (back light off); ~1 hr (back light on)
– Lithium battery preserves memory contents during battery low condition; additional battery preserves memory contents during battery change.

Size – 2.9 x 5.4 x 0.7 in. (with keyboard hidden)
Weight – 6.8 oz.
Bundled accessories: USB connected cradle; AC adapter

11.2.   Software :

Sharp Zaurus SL-5500 has the following software specifications [19] :

Operating system – based on Lineo's Embedix Embedded Linux with Linux Kernel 2.4.x

Java runtime environment – Insignia's Jeode PDA Edition (a Sun-authorized Virtual Machine that is compatible with the Personal Java Specification)

GUI – based on Trolltech's Qt/Embedded

Browser – Opera

Application software – based on Trolltech's Qtopia PDA suite: includes productivity suite (todo list, calendar, calculator, address book, text editor), entertainment package (mpeg player, image viewer, games), internet package (email client, web access), utilities (screen calibration, backlight control, app installer, network setup, I/O controls).

12. Flashing Zaurus ROM

Inorder to flash new operating system (OS) in the Zaurus ROM the following steps need to be followed [20] :

1. Plug in the cord to the Zaurus. You will find a hole for the cord at the bottom right of the Zaurus when held face up.

 2. Charge the machine for 10 minutes.

3. Flip over the Zaurus so that it is face down. Open the back cover. (There is a gray switch in the bottom right corner that is to the right of the words "NORMAL OPERATION" and "REPLACE BATTERY".) Locate the RESET button but do NOT press it. It is in the square indentation to the left of a gray button, just below the battery, and above the words "NORMAL OPERATION" and "REPLACE BATTERY".

4. Insert the compact flash card (which contains the image file) at the top of the Zaurus.

5. Turn over the Zaurus so that it is face up. Pull down on the bottom half of the top half of the Zaurus to reveal a keyboard.

6. Simultaneously press the RESET button and the letters C and D in the keyboard to reset the program. This action will cause a green light (just above the envelope icon) to turn on the face-up side of the Zaurus just below the display screen (monitor).

7. Wait for few minutes until the green light turns off.

8. Replace the back cover of the Zaurus. Using the gray switch, lock the back cover in place.

9. Press the power button on the front cover to see the new program reloading. To make sure that you flashed correctly, once you turn the Zaurus on, it should say calibrating. To finish, simply tap the four corners of the screen as instructed.

Note: We flashed Zaurus ROM with new ROM image, which includes OpenZaurus 3.5.2 (with Linux Kernel 2.4.18-rmk7-pxa3-embedix-021129 kernel) as its Operating environment and Opie 1.1.7 as GUI.

13. OpenZaurus:

The original purpose of the OpenZaurus Project was to create a ROM image (Kernel + Root Filesystem). The intention of the distribution is to continue to produce a solid software distribution for the Sharp Zaurus SL-5000d (eventually SL-5500 as an alternative to the default distribution from Sharp. Binary compatibility will be attempted to retain both to and from the Sharp ROMs. This distribution is also targetted to be a bit closer to the developer community, and will do its best to take advantage of the closeness, by making use of the assistance of any willing developers [21].

OpenZaurus provides 16MB of built-in Flash separate from the 64MB of memory, which is split into the SL-5500, 32MB for storage and 32MB for heap. The heap memory also can be written to, allowing you to add and remove applications that you can't be manipulated in the Sharp ROM. The most important thing is that OpenZaurus offers freedom of choice [22].

14. OPIE:

OPIE, whose full form is the **O**pen **P**almtop **I**ntegrated **E**nvironment is a 'fork' of Qtopia environment developed by Trolltech. It is a completely Open Source graphical user environment for PDA's and other devices running Linux. The maintainance is done by a group of people from all over the world, convinced by the Open Source Policy. Opie uses Qtopia greatly in an extended form thus establishing itself as the most sophisticated free and open graphical user interface for Linux based embedded devices and PDAs.

Opie is furnished with a sophisticated personal information (PIM) framework and at the same time possesses several productivity applications. It also has extended multimedia capabilities, dcument model, networking and communication tools, multi-language support for more than a dozen languages etc. Opie efficiently interacts with lots of devices from cell phones to server back ends. Opie is based on common industry standards like XML, Obex, IrDa etc. [23].

14.1. Qt/Embedded:

The Qt/Embedded product provides all that is required to create the graphical user interfaces for embedded Qtopia.

Qt/Embedded is installable and runnable with a very small memory on any device running embedded Linux at the same time not using X11. Qt/Embedded has the same API as the popularly used Qt/Windows and Qt/X11 versions. So this saves a lot of time incase of moving applications developed in a particular desktop environment . Additionally, only recompiling is required.

Qt is undoubtedly the most popular GUI toolkits in the world. It is preferred by programmers for its compact code, powerful API, ease of use and the excellent support [25].

14.2. Opie compilation for x86 and ARM:

Below is a list of software tools required to compile OPIE [26] :

1. uic-qt2
2. qvfb-qt2
3. qt-embedded-2.3.10-free.tar.gz
4. Stable Opie Source

14.3. Additional tools required to compile OPIE for ARM processor:

To compile OPIE source under Linux for targeted device, it need a cross compiler (toolchain). The toolchain required for the cross compilations are [27] :

1. Cross Compiler (gcc)
2. Libraries (glibc)
3. Utility Tools (other than gcc)
4. Header Files

So, we downloaded the following toolchain for Sharp Zaurus

1. binutils-cross-arm-2.11.2-0.i386
2. gcc-cross-sa1100-2.95.2-0.i386
3. glibc-arm-2.2.2-0.i386
4.linux-headers-arm-sa1100-2.4.6-3.i386

14.4. Additional libraries and header files Required to compile OPIE for ARM processor [28] :

1. {libjpeg62,libjpeg62-dev}_6b-5_arm.deb
2. {libfreetype6,libfreetype6-dev}_2.0.9-1_arm.deb

3. {zlib1g,zlib1g-dev}_1.1.4-1.0woody0_arm.deb

4. {libpcap0.7,libpcap0.7-dev}_0.7.2-7_arm.deb

5. {libpng3,libpng-dev}_1.2.1-1.1.woody.9_arm.deb

6. {libbluetooth1,libbluetooth1-dev}_2.11-1_arm.deb

7. {libgcc1_3.0.4-7_arm.deb

8. {libpcsclite1,libpcsclite-dev}_1.2.9-beta6-1_arm.deb

9. flex_2.5.4a-24_arm.deb

10. libpam0g-dev_0.76-9_arm_for_Opie.tgz

14.5. Additional Tools Required to compile OPIE for x86 processor:

1. {libbluetooth1,libbluetooth1-dev}_2.15-2_i386.deb

2. {libfreetype6, libfreetype6-dev}_2.0.9-1_i386.deb

3. {libjpeg62, libjpeg62-dev}_6b-5_i386.deb

4. libpam-unix2_1.25-1_i386.deb

5. {libpcsclite0,libpcsclite-dev}_1.0.2.beta5-1_i386.deb

6. {libpng3, libpng-dev}_1.2.1-1.1.woody.9_i386.deb

7. pkg-config_0.20-1_i386.deb

8. zlib1g_1.1.4-1.0woody0_i386.deb

9. zlib1g-dev_1.2.3-9_i386.deb

The compilation procedure for both x86 machine and ARM machine is included in the **APPENDIX** section.

14.6. Translation of OPIE

The Opie-Project tries to offer the support for as many languages as possible.To ensure that Opie can be used by as many people as possible the Opie-project aims to translate OPIE in as many languages as possible. Since OPIE is an evolving project, OPIE files and consequently files to be translated keep on adding and hence the translation would need to be constantly updated.

In order to translate Opie files, an editor (translation tool) is required to edit the translation file. It could be any tool preferably supporting UTF8. The preferred editors are QtLinguist, KBabel etc.

14.7. Steps for translating OPIE into the native language [29]

- To create a new language entry, a folder of any name (language name) should be added in the $OPIEDIR/i18n/ directory.

- After the entry of the new language, create a file opie-i18n-desired_language_name.control in the directory $OPIEDIR/i18n/

- Then create .directory file, which includes the name of the desired language, and create the array of its name to be appeared on other different language attributes.

- Run the command opie-lupdate or add .ts files manually.

  o Opie-lupdate - opie-lupdate is an specialized version of Qt3.1 lupdate.

    ▪ You do not need TRANSLATIONS in your .pro anymore. Opie
    ▪ lupdate reads the language list from $OPIEDIR/i18n and then generates the .ts files.

  o opie-lrelease - Also deprecates the TRANSLATIONS attribute inside the .pro files

- The translation sources are located in the *.ts files. *.qm files are generated during another part that needs to be translated in the Desktop and .directory files, found in the OPIE directory tree.

- After translation in the respective language is complete, Font containing that particular Unicode characters need to be installed or created for OPIE.

  Font format of PDA is .qpf (packed font). The program that is used to create these fonts is "makeqpf". The tool "makeqpf" could be used to change the font in TTF & BDF format to PDA compatible fonts (QPF).

  The method to install new font in PDA is included in the **APPENDIX** section

14.8. OPIE file format

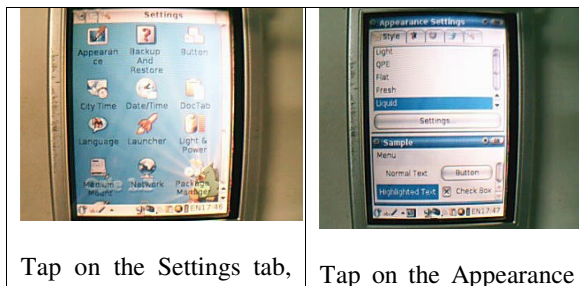The OPIE files for translation come under the *.ts format. A typical example of the .ts file format is presented below:

```
<message>
    <source>Application</source>
    <translation>अनुप्रयोग</translation>
</message>
<message>
    <source>Settings</source>
    <translation>सेटिङगहरु</translation>
</message>
<message>
    <source>Pim</source>
    <translation type="unfinished"></translation>
</message>
```

To briefly explain the format of the .ts file, it is a simple markup file with few important tags like "source" and "translation". The string to be translated is enclosed within the tags "source" whereas the translation appears in between the tags "translation". If some strings are untranslated, an additional attribute type ="unfinished" appears in the tag "translation" as evident above in the example. Translation tools like Kbabel facilitate an easy mode of translating the strings by grouping the strings to be translated in the upper part of the window and the actual translation string input in the lower half. Both the strings to be translated and the translation for the strings may be navigated with the help of the available navigation tools.
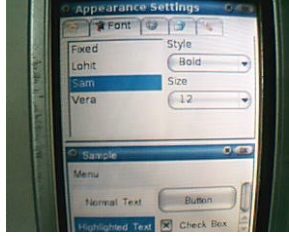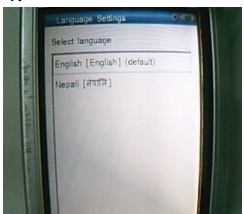
14.9. Getting the OPIE interface in Nepali language

Once the translation of OPIE files is complete, the following steps are required to get the OPIE interface in the Nepali language.

 Tap on the Settings tab,	 Tap on the Appearance

you'll get the screen as shown.	Icon, you'll get the screen as shown.
 Tap on the Font tab, you'll get the screen as shown. Then tap on the 'Sam' font and select the appropriate Size and Style and then tap on 'OK' button on the right top of the screen. This changes the whole interface to be appeared on 'Sam' font.	4. Again go to 'Settings' tab and Tap on the 'Language' Icon, you'll get the screen as shown. Now, select 'Nepali' language and tap on 'OK' button on the right top of the screen. After few seconds you'll get the interface changed into 'Nepali' language.

Note: 'Sam' is the font created by the PDA localization Nepali team (by modifying 'Mangal' font and adding English characters onto it), which consists of all the English characters, Basic Symbols and Nepali Characters.

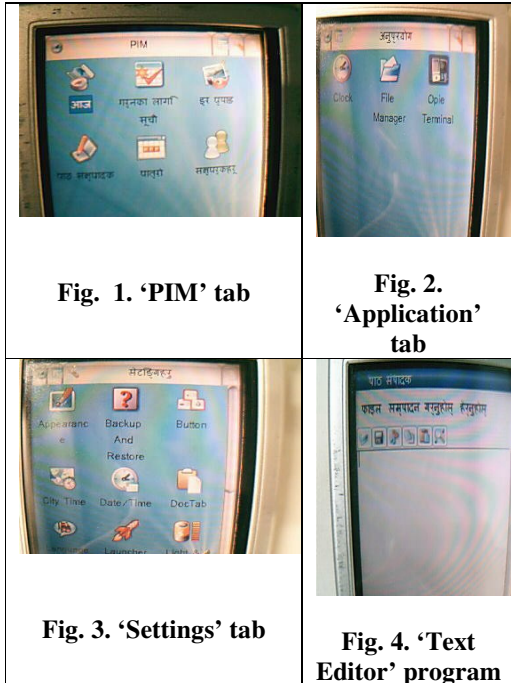**Some screenshots of OPIE interface in Nepali Language**

Fig. 1. 'PIM' tab

Fig. 2. 'Application' tab

Fig. 3. 'Settings' tab

Fig. 4. 'Text Editor' program

15. Problems

Proper rendering of the Nepali script was not achieved. This turns out to be the major limitation of the project. Although attempts were made to correct the existing rendering problem by contacting the OPIE developers community, concrete results were not achieved.
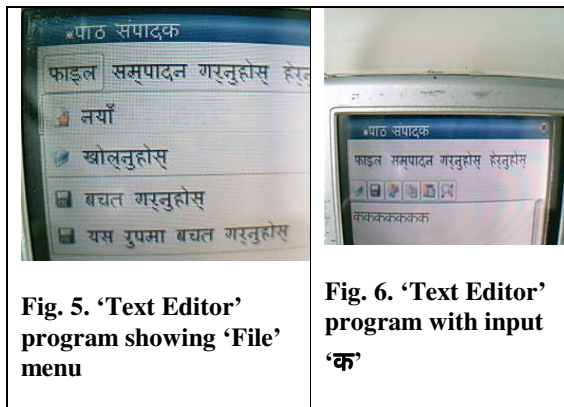

Fig. 5. 'Text Editor' program showing 'File' menu

Fig. 6. 'Text Editor' program with input 'क'

Customization of the existing input schemes of the

16. PDA into Nepali

After the relatively successful customization of the PDA interface into Nepali, efforts were made in the customisation of the existing input schemes for the PDA. This involved customizing the virtual keyboard, an input facility existing in the PDA and doing some basic research on the handwriting recognition part concerning the input through the stylus.

17. Brief overview of the input schemes available

17.1. Keypad

Every Pda has a reduced keyboard for the input. The user can directly use the keys to provide the input to the PDA. It is just as the layout for the normal keyboards but in the reduced format.

17.2. Multikey

MultiKey is like the virtual keyboard to the PDA, which appears at the bottom of the screen with all the corresponding keys associated with the specific letters, functions and numbers.This is the input scheme that provides the virtual keyboards to give the inputs. A small icon is present at the bottom left corner and it has a scheme for changing the type of the input. One can select multikey environment and provide input using virtual keyboard layout displayed on the screen and press on the icons for the keys on the screen using the pressure pen. Further illustrations and screen shots will appear in the later section of this documentation.

17.3. Pickboard

It is quite similar to to the multikey environment and providing input in a quite similar manner to multikey (i.e. using pressure pen).

17.4. Handwriting

The **Handwriting** input mode recognizes stylus strokes and converts them into letters, numbers and symbols. The strokes are made with the help of the pressure pen.

17.5. Customization of Multikey environment

"Multikey" input method is a purely customizable input method where every key of the virtual keyboard can be mapped with the corresponding characters,

functions and numbers to that of Nepali Unicode Romanized Layout.

Fig.7. Nepali Unicode Keyboard Layout (Romanized) for normal keyboard.

As evident from the discussion above, the same layout presented above has been implemented in the "Multikey" input environment.

Steps for enabling the native language input (Nepali) [33]

1. Copy en.keymap to be template for your new keyboard.
cp /opt/QtPalmtop/share/multikey/en.keymap
/opt/QtPalmtop/share/multikey/np.keymap
np.keymap shall be changed to your own country(nepali in our case).

2. Edit content of np.keymap

```
# comment lines must start with a '#' (for now...)
# order is: row qcode unicode length blah
#
# please don't write anything between a key definition
and its xpm (except for s

title = English
sw = EN
```

i. Change Title and sw to your own language. (sw is shown at taskbar).

In our case we made Nepali keyboard as

```
title = Nepali
sw = NP
```

ii. Then change the contents as wish.

```
1 0      0x60   2 # but not after xpm images...
k?want after the last element
1 0      0x31   2
1 0      0x32   2
1 0      0x33   2
1 0      0x34   2
1 0      0x35   2
1 0      0x36   2
1 0      0x37   2
1 0      0x38   2
1 0      0x39   2
1 0      0x30   2
1 0      0x2d   2
1 0      0x3d   2
1 0x1003   0      2
```

1 0 means first row of keyboard. you and see that it ascii code of `1234567890-
Likewise 2 0 is second row 3 0 is third row 4 0 is forth row

You now then have to know your language ascii or unicode. Then replace it as you want.

Ascii is number that less than 0xff. Unicode is number that have more than 2 digits such as 0x967

Feel free to change trial and error until you satisfy with new keyboard.

iii. After you finish keyboard with out shift. Now you have to modify to have shift key.

```
# shift table
# add xpm support for these keys too?

# unshifed shifted
0x60  0x7e
0xe45  0x2b
```

0x2f 0xe51
0x2d 0xe52
0xe20 0xe53

iv. Just change the number behind first number to get shift key.

Here is the small piece of the file illustrating above facts:

This is for the first row ie for १२३४५……

```
1 0     0x967   2
1 0     0x968   2
1 0     0x969   2
1 0     0x96A   2
1 0     0x96B   2
1 0     0x96C   2
1 0     0x96D   2
1 0     0x96E   2
1 0     0x96F   2
1 0     0x966   2
1 0     0x2d    2
1 0     0x3d    2
1 0x1003  0     2
```

Now here is the small sample for the shifted keys.

```
# unshifted shifted
0x91F 0x921
0x94C 0x914
0x947 0x948
0x930 0x943
0x924 0x925
0x92F 0x91E
0x941 0x942
0x93F 0x941
0x94b 0x913
0x92A 0x92B
0x907 0x908
0x90F 0x910
0x950 0x903  … similarly you may proceed for other
keys.
```

v. What if the character does not get displayed on the keyboard?

There might be the case where the character associated with the keys of keyboard is not actually displayed on the keyboard i.e. key appears blank.

In that case, OPIE offers a feasible way to represent the corresponding key to its mapping using the drawing technique in terms of matrix representation. In drawing technique, one needs to edit the file "np.keymap" for Nepali Input to represent how the display exactly looks like for the particular keys.

Eg:

```
1 0x1000   0     2

  "9 9 2 1"
  ". c None"
  "a c #000000"
  "........."
  "........."
  ".aa.aa.aa"
  ".a..a..a."
  ".aa.aa.a."
  ".a...a.a."
  ".aa.aa.aa"
  "........."
  "........."
```

Here, the first line indicates the position and the Unicode value of the corresponding key. '1' in first line indicates that the position is "Row-1", and the Unicode value is "0x1000" which represent "ESC" character. In this case, as the key "ESC" does not have any representation on the Unicode mapping to actually represent it. So, we need to draw its representation which indicates it as "ESC" characters i.e. we need to draw the "ESC" on the key that need to be displayed.

"9 9 2 1": '9' represent number of rows
          '9' represent number of columns

". c None" : this indicates '.' is the blank or simply of color white.

"a c #000000" : this indicates 'a' is colored to black

This now forms 9x9 matrix, and a complete display is represented in this representation, where the pattern of 'a' colors the pixels into black on the background of white i.e. '.'. So, inorder to draw a desired character or words that is to be displayed in the keyboard, one needs to manipulate the pattern of 'a' and '.' to draw them in a complete matrix representation.

Using this technique, we represented many characters to be displayed in the keyboard.

ॊ is represented as below:

```
2 0      0x947  2

"9 9 2 1"
  ". c None"
  "a c #000000"

  ".a......."
  "..a......"
  "...a....."
  "...aaa..."
  "........."
  "........."
  "........."
  "........."
  "........."
```

Where, 0x947 is the Unicode value for ॊ.

18. Customization of the existing Handwriting Recognition feature on PDA for Nepali

18.1. Handwriting

The **Handwriting** input mode recognizes stylus strokes and converts them into letters, numbers and symbols. To use this mode, you will need to adapt your **handwriting** style slightly to the way Opie expects characters to be drawn, and you enter characters on the spot (rather than moving your position as you write, as you do when you write on paper).

The **handwriting** area is split up into three parts from left to right - upper case letters (marked ABC), lower case letters (marked abc), and numbers and symbols (marked 123). There are four buttons to the right of the **handwriting** area - backspace, enter, help, and settings (pen icon).
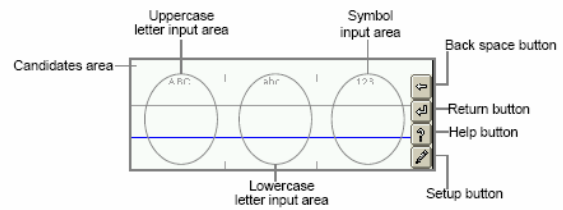
Figure 8. Input Areas

18.2. Training

A good way to learn how to use the **handwriting** input mode is to use the trainer. To access the trainer, tap on the help button (? icon) and select the Trainer tab. Select each character in turn - Opie will show you an animation of how to draw the character on the left. Try writing the character in the test area on the right, and Opie will tell you how accurately it is able to work out which character you just drew.

You will get the best results if you draw characters similarly to the way they are drawn in the animated example.
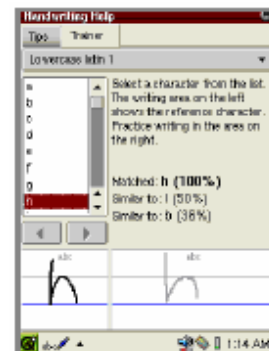

Figure 9. Training

18.3. Settings

Other than the Customize tab, there are just two settings you can change here - the multi-stroke character timeout (the time allowed between strokes to enter a multi-stroke character, eg. f); and the splitting of the **handwriting** area - you may choose to have a two-part area (toggling between upper/lower case in the letter side).
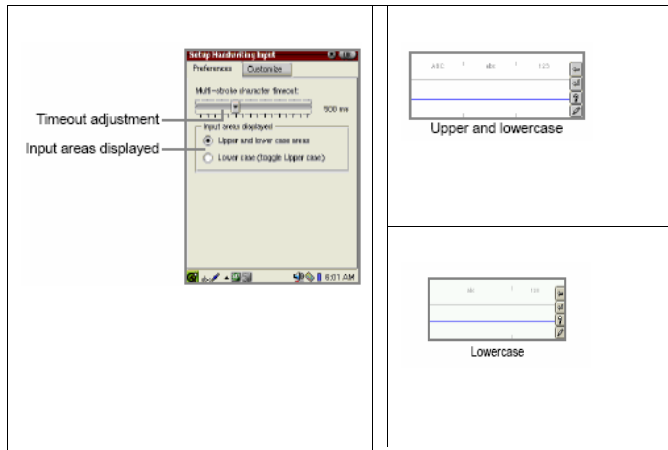
Figure 10. Timeout adjustment (Multi-stroke character timeout)

18.4. Customize

If you wish, you may edit the strokes used to enter text in the **handwriting** input method on the Customize tab. Choose the character set you wish to edit from the drop-down list. You can select a character from the list, and see the strokes associated with it using the arrow button (if there is more than one). Use the actions below to perform your editing.

18.5. Adding a new stroke

To add a new stroke to an existing character (a different way of entering the same character), draw it carefully in the entry area at the bottom of the screen, and then press "Add". If you make a mistake, press the Clear button and start again.

18.6. Adding a new character

To add a new character (one not already in the list), draw it carefully in the entry area at the bottom of the screen, and then press "New...". If you make a mistake, press the Clear button and start again.

18.7. Removing strokes
To remove a stroke from a character, use the arrow buttons to select press the Remove button

18.8. Resetting a character

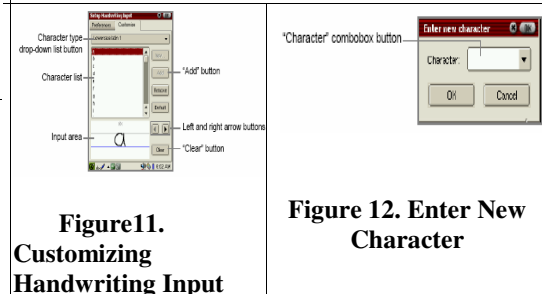To reset the selected character's strokes back to the efaults, press the Default button.

Figure11. Customizing Handwriting Input

Figure 12. Enter New Character

19. Research on handwriting recognition

After having worked with the training of basic Nepali characters as input with the help of stylus and pressure-sensitive pad of the PDA,
Optical Character Recognition over the years has been one of the hottest topics for research in the field of computing. Various researches have been performed and various outcomes have resulted. We find different algorithm in the fields that compete against each other in terms of efficiency and reliability.

Handwriting recognition problems are categorized into two types, viz., online and offline. Incase of online recognition systems, a pressure-sensitive pad records the pen's pressure and velocity (for example, a PDA). On the other hand, in offline recognition, the system input is a digital image of handwritten letters and numbers [30] .

19.1. On-line Character Recognition

Alphabet character recognition takes an input character and assigns it as one of the possible output classes. The process comprises two general stages: *feature selection* and *classification*. Feature selection is considered to be a very critical stage to the whole process as correct recognition is not possible from poorly selected features. The task of feature selection is often done by the researcher manually, but a neural network approach automatically extracts the relevant features [31] .
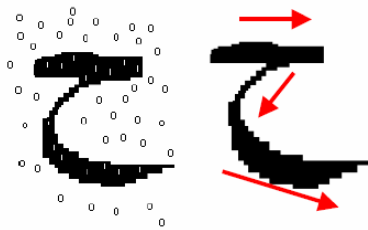
Figure 13. Examples of off-line(left) and on-line(right) handwriting inputs

19.2. On-line vs. Off-line

A raster image is taken from a scanner, digital camera or other digital input source incase of offline character recogniton. The input image is binarized (digitized) using a threshold technique (image pixels are either on (1) or off (0)). Rest of the preprocessing is similar to the on-line version with two prime differences: Off-line processing deals with scanned image of complete written characters. Off-line inputs do not have temporal information associated with the image (relationships between pixels or the order of the creation of strokes cannot be inferred).

On the other hand, on-line character recognition accepts (x,y) co-ordinate pairs from an electronic pen touching a pressure-sensitive digital tablet. In this case, processing takes place in real-time. Relationships between pixels and strokes also are taken into account [31] .
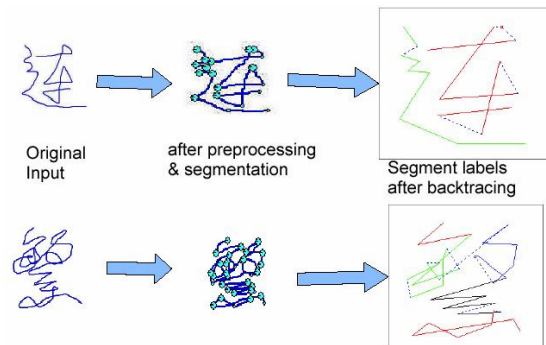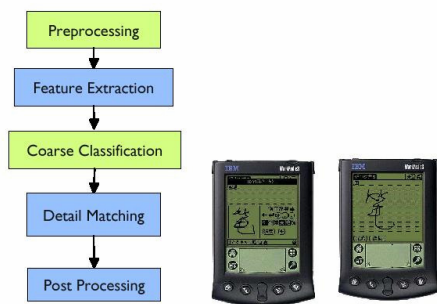
A general recognition procedure

Figure 14. Result of each processing for the given input

19.3. Feature Extraction

A feature point is a point of human interest in an image (intersection between two lines, corner, dot surrounded by space etc.) The relationship between different strokes is defined by such points. These relationships are used for character identification and the feature points are used for the task.

The feature points are extracted by using certain algorithms. For example, consider an eight by eight character consisting of only sixty-four pixels. It is viable to simply loop through the entire character and examine each pixel in turn. If a pixel is on, its eight neighbors are checked. Since each neighbor can also only be on or off, there are 256 possible combinations of neighborhoods. Out of these 256, fifty-eight are found to represent significant feature points in a fairly unambiguous manner. Extracting feature points thus reduces to calculating a number between zero and 256 to describe a pixel's neighborhood and then comparing that number against a table of known feature points. This method does not always catch every feature point (some can only be seen in a larger context) but it catches the majority.

Extraction of feature points alone cannot help identify characters. In this regard, a database of characters and their associated feature points is a must. This will help comparing the results of the extracted features of an unkonown character against real characters [32].
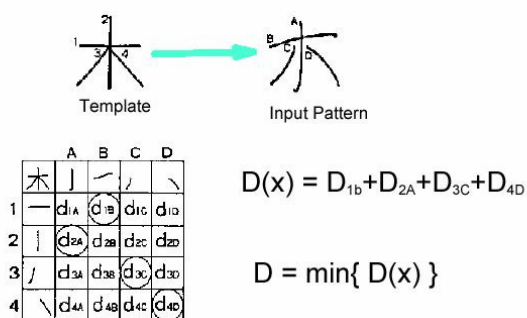
An Example of Stroke correspondance

$$D(x) = D_{1b} + D_{2A} + D_{3C} + D_{4D}$$

$$D = \min\{ D(x) \}$$

Figure 15. Stroke Correspondence

These are the basic procedures adapted in handheld devices to recognize the handwriting input.

20. Conclusion

The scope of the PDA Localization project undertaken by the Nepal component of the PAN localization project was initially only to localize the basic PDA interfaces. Hence the customization of the input parts followed by a small research on handwriting recognition was something done in addition. All the processes involved in PDA localization starting from the study of the existing PDA technology to the choice of the operating system and the model for customization, localization and translation procedures thoroughly covered during the research and development of PDA localization have been duly documented in the report. Technical details on some specific portions like compilation and building are presented in the Appendix section.

21. References

[1] http://electronics.howstuffworks.com/pda.htm/printable

[2] http://www.howstuffworks.com/pda.htm/printable

[3] http://www.bollywoodstuff.info/index2.html

[4] "Cell phone: Cell Phone, Mobile Phone Reviews, Wireless Phones - CNET" http://reviews.cnet.com/Handhelds/2001-3127_7-0.html?tag=cnetfd.glnav

[5] "ACESS" http://www.palmsource.com/palmos/

[6] "Symbioforge" http://www.symbioforge.com/feii/

[7] "PiLoc Download Wizard" http://piloc.penreader.com

[8] http://TwinyPalm.com

[9] http://www.w3c.org/TR/1999/REC-html401-19991224/loose.dtd

[10] "Sharp Zaurus SL-5600 Personal Mobile Tool" http://www.linuxjournal.com/node/6792/print

[11] "Code Less Create More" http://www.trolltech.com

[12] http://ftp.trolltech.com/products/qtopia/index.html?cid=21

[13] http://www.linuxdevcenter.com/pub/a/linux/2004/01/29/zaurus.html

[14] http://schockwellenreiter.server-wg.de/blog/daylist_html?year=2004&month=2&day=6

[15] http://xiongxiong.com/htm/Developer/DevelopNews/2004225C104254.html

[16] http://www.lisoleg.net/cgi-bin/lisoleg.pl?view=news/messages/msg.3541

[17] http://www.kaii.info/BusinessLine.htm?hl=en&lr=&ie=UTF-8&num=10&q=related:www.blonnet.com/2002/08/20/stories/2002082001210700.htm

[18] http://www.thehindubusinessline.com/2002/08/20/stories/2002082001210700.htm

[19] http://216.218.185.154/articles/AT2134869242.html

[20] http://dev-bywater.media.mit.edu/wiki/borglab/Flashing_the_Zaurus_ROM

[21] http://www.linuxdevices.com/news/NS2605436593.html

[22] http://new.linuxjournal.com/node/7866

[23] http://opie.handhelds.org/overview.php

[24] http://theqtopian.net/Xoops/modules/news/print.php?storyid=7&

[25] http://216.218.185.154/cgi-bin/printerfriendly.cgi?id=PD9968824320

[26] http://www.bigbridgezau.sakura.ne.jp/wiki.cgi?page=Qtopia-free-1.7.1%A5%D3%A5%EB%A5%C9%B4%C4%B6%AD%B9%BD%C3%DB

[27] http://www.uv-ac.de/opiedev/opiedev-4.html

[28] http://people.via.ecp.fr/~clem/nist/qt-notes.php

[29] http://en.pdamobiz.com/en/forum/forum_posts.asp?TID=200&PN=1&get=last

[30] http://wearcam.org/ieeecomputer/0297abs.htm

[31] http://users.cs.dal.ca/~mheywood/Reports/TKlassen.pdf

[32] http://www.ccs.neu.edu/home/feneric/charrec.html

[33] http://en.pdamobiz.com/en/forum/forum_posts.asp?TID=171&PN=1&get=last

[34] http://people.via.ecp.fr/~clem/nist/qt-notes.php

[35] "Guide to the Qt Translation Tools" http://doc.trolltech.com/qtopia2.1/html/linguist-manual.html

[36] "Zarus Software Index" http://www.killefiz.de/zaurus/showapps.php?cat=6

[37] "News-Opie" http://opie.handhelds.org/cgi-bin/moin.cgi/

[38] http://www.zaurus.com/dev/tools/other.htm

[39] "CreateFontsForOpie-Opie"http://opie.handhelds.org/cgi-bin/moin.cgi/CreateFontsForOpie

[40] "Open Palmtop Integrated Environment (Opie)" http://opie.handhelds.org/overview.php

[41] "Opie development-PDA development for Compaq IPAQ and Sharp ZAURUS: Getting started" http://www.uv-ac.de/opiedev/opiedev-4.html

[42] "Heaton Research: Java Tutorials, Neural Networks and More" http://www.heatonresearch.com

22. APPENDIX

1. Compile OPIE from source for ARM and x 86 platforms

After qt-embedded and opie source had been downloaded. It was extracted with appropriate command through the terminal.

2. Set up build environment

Build environment were then set after qt-embedded and opie source had been extracted.

cd ~/opie
export OPIEDIR="$(pwd)"

cd ~/qt-embedded
export QTDIR="$(pwd)"

3. uic compiler

In order to compile Opie, QT/Embedded need to be built. But in order to compile QT/Embedded, QT/X11 uic compiler is needed. That compiler is needed to "translate" UI files (user interface files, created with qt2-designer, the layout designer from TrollTech) to cpp files.

The easiest way is to download this file (statically compiled) from http://vanille.de/tools/uic-qt2

Now its permissions need to be changed to make it executable.

chmod u+rx uic-qt2

4. qvfb

To run Opie on x86 linux plateform, qvfb (Qt Virtual Frame Buffer) is needed. It could be downloaded , as for uic, from http://vanille.de/tools/qvfb-qt2

Now its permissions need to be changed to make it executable.

chmod u+rx qvfb-qt2

5. Link/move QT/X11 tools

uic-qt2 is now linked to the QT/Embedded binary folder

cd $QTDIR
mkdir bin
ln -s path_of_uic-qt2 bin/uic

same for qvfb

ln -s path_of_qvfb-qt2 bin/qvfb

6. Patch and stuff
linked to QTE config-file

ln -s $OPIEDIR/qt/qconfig-qpe.h src/tools/

patch Qt/Embedded to work correctly with Opie and remove some Qt/Embedded errors

patch -p1 < $OPIEDIR/qt/qt-2.3.10.patch/qte-2.3.10-all.patch

To compile OPIE for ARM processor

To compile OPIE source under Linux for targeted device, it need a cross compiler (toolchain). The toolchain required for the cross compilations are:

- binutils-cross-arm-2.11.2-0.i386
- gcc-cross-sa1100-2.95.2-0.i386
- glibc-arm-2.2.2-0.i386
- linux-headers-arm-sa1100-2.4.6-3.i386

These are RPM files, so could be easily installed. Additional libraries and header files Required to compile OPIE for ARM processor:

- {libjpeg62,libjpeg62-dev}_6b-5_arm.deb
- {libfreetype6,libfreetype6-dev}_2.0.9-1_arm.deb
- zlib1g,zlib1g-dev}_1.1.4-1.0woody0_arm.deb
- {libpcap0.7,libpcap0.7-dev}_0.7.2-7_arm.deb
- {libpng3,libpng-dev}_1.2.1-1.1.woody.9_arm.deb
- {libbluetooth1,libbluetooth1-dev}_2.11-1_arm.deb
- {libgcc1_3.0.4-7_arm.deb
- {libpcsclite1,libpcsclite-dev}_1.2.9-beta6-1_arm.deb
- flex_2.5.4a-24_arm.deb
- libpam0g-dev_0.76-9_arm_for_Opie.tgz

are then extracted and the library files are then placed on the *arm-linux/lib* directory of the cross-compiler directory and the header files are then placed on the *arm-linux/include* directory of the cross-compiler directory.

Now the path is needed to be set correctly.

export LD_LIBRARY_PATH=directory of cross-compiler/arm-linux/lib:$LD_LIBRARY_PATH
export PATH=directory of cross-compiler/arm-linux/bin:$PATH

22.1. Building Qt for Opie (ARM target)

./configure -qconfig qpe -depths 4,16,24 -xplatform linux-sharp-g++ -no-qvfb -system-jpeg -system-libpng -gif -system-zlib -vnc -no-xft

make

Once Qt is built, set the PATH to it:

export PATH=$QTDIR/bin:$PATH

22.2. Configure and make opie

cd $OPIEDIR

make clean
make menuconfig //select or deselect the application to be installed
make

22.3. To compile OPIE for x86 processor

Additional libraries and header files required to compile OPIE for x86 processor:

- {libbluetooth1,libbluetooth1-dev}_2.15-2_i386.deb
- {libfreetype6, libfreetype6-dev}_2.0.9-1_i386.deb
- {libjpeg62, libjpeg62-dev}_6b-5_i386.deb
- libpam-unix2_1.25-1_i386.deb
- {libpcsclite0,libpcsclite-dev}_1.0.2.beta5-1_i386.deb
- {libpng3, libpng-dev}_1.2.1-1.1.woody.9_i386.deb
- pkg-config_0.20-1_i386.deb
- zlib1g_1.1.4-1.0woody0_i386.deb
- zlib1g-dev_1.2.3-9_i386.deb

are then extracted and the library files are then placed on the *lib* directory of the /usr directory and the header files are then placed on the *include* directory of the /usr directory.

22.3.1. Building Qt for Opie (x86 target)

./configure -qconfig qpe -depths 4,16,24,32 -system-jpeg -system-libpng -system-zlib -no-xft -qvfb

make

Once Qt is built, set the PATH to it:

export PATH=$QTDIR/bin:$PATH

22.3.2. Configure and make opie

cd $OPIEDIR
make clean
make menuconfig //select or deselect the application to be installed
make

22.3.3. Setup opie

```
export QTDIR=~/qt-embedded
export OPIEDIR=~/opie
export
LD_LIBRARY_PATH=$QTDIR/lib:$OPIEDIR/lib
export PATH=$QTDIR/bin:$OPIEDIR/bin:$PATH
export QWS_DISPLAY=QVFb:0

qvfb
qpe
```

- *# update-qtfontdir*
- Change default font at, Settings > Appearance > Font > Tahoma > ok

Now your opie font change from Fixed to Tahoma...

22.3.4. Methods for creating and installing new fonts on PDA [33] :

- makeqpf was downloaded:
  http://moria.ionkov.net/zaurus/makeqpf/makeqpf-arm.zip
- unzipped we get file makeqpf-arm
- copy to /usr/bin
- *# cp /mnt/card/makeqpf-arm /usr/bin*
- *# chmod +x /usr/bin/makeqpf-arm*
- Get ttf font
- Copy that font to /opt/QtPalmtop/lib/fonts
- example: *# cp /mnt/card/tahoma.ttf /opt/QtPalmtop/lib/fonts*
- Modify file: fontdir to have contents as
- tahoma tahoma.ttf FT n 50 0 su 70,80,100,120,140,160
- Some Meaning:
- FT is for ttf file, BDF is for bdf file
- n is not italic (y is italic)
- 50 is normal, 75 is bold
- 70,80,100.... is the font size start from 7,8,10,12,14,16
- Run following command (some error message but no problem)
- *# makeqpf-arm -A -f /opt/QtPalmtop/lib/fonts/fontdir*
- *# makeqpf-arm -A -display Transformed:Rot90 -f /opt/QtPalmtop/lib/fonts/fontdir*
- *# makeqpf-arm -A -display Transformed:Rot180 -f /opt/QtPalmtop/lib/fonts/fontdir*
- *# makeqpf-arm -A -display Transformed:Rot270 -f /opt/QtPalmtop/lib/fonts/fontdir*
- now restart opie, or
-
  8. update fontdir file for new qpf files