Malaysian Journal of Computer Science, Vol. 10 No. 2, December 1997, pp. 38-44

LOCALISING A SPREADSHEET: AN IBAN EXAMPLE

Alvin Yeo
Department of Computer Science
University of Waikato
Private Bag 3105, Hamilton
New Zealand
Tel: 64 7-856 2889
Fax: 64 7-838 4155
email: awy@cs.waikato.ac.nz

Robert Barbour
Science, Mathematics and Technology Education
Research Centre
University of Waikato
Hamilton, New Zealand
Tel.: 64 7-838 4403
Fax: 64 7-838 4272
email: r.barbour@waikato.ac.nz

ABSTRACT

Presents an example of the localisation of a spreadsheet from English to Iban. The process by which the localisation has been carried out can be used as a framework for the localisation of software to languages of small ethnic minorities. Some problems faced during the localisation process are also discussed.

Keywords: ***Language translation, software localisation, Iban spreadsheet.***

1.0 INTRODUCTION

Most commercial software production is dominated by products from computer firms in the United States (US). These firms not only produce software in English but also other languages. For example, the Apple Macintosh operating system is available in 30 major languages [1]. However, localisation to these languages probably occurred because it was economically viable to do so. Languages with relatively fewer speakers, like Mäori (an indigenous ethnic group in New Zealand) or Iban (an indigenous ethnic group in Sarawak, Malaysia), are unlikely to feature in localised software.

Marketing people say there is no profit in localising software for minorities and that those people who wish to use their software should learn English. We strongly disagree with this viewpoint. We hold the view that having access to one's own culture in an indigenous language is an essential human right and critical for sustaining a culture. If a language ceases to exist, a major component of the culture also disappears. With it, potential perspectives or solutions to future human problems may also disappear. Furthermore, Griffiths et al. [2] report that people learn and progress faster using software with an interface in their native language.

There are many factors which need addressing in the localisation of a software. These factors include the translation of languages, icons, collating order, date and time formats. In this article, we will focus on the translation of an English text user interface to Iban[1]. The application program discussed is a spreadsheet. This application was selected because managing money in an indigenous language is one way of reducing the hegemony of English in non-western cultures. We chose an available DOS based spreadsheet because we were given access to source code by Borland International for the purposes of the research. No software house with a more current GUI based spreadsheet interface would provide source code for our experiments. Items requiring translation are outlined in Section 2. Possible approaches for translating various components of the software are described in Section 3. We present a summary in Section 4 and areas for further work in Section 5.

2.0 WHAT NEEDS TRANSLATING?

Internationalised software is designed so that many target languages can be accommodated within a single application. Localisation is the process of selecting a target market and modifying the application by translating key language components for that market. These components can be categorised into software and non-software components. The software components consist of the visual elements on screen. The components include text strings for menus, dialogue boxes, help files and error messages. Uren et al. [4] list the following non-software components for translation:

- Manuals
- Quick reference cards
- README files
- Messages in on-line Help
- Boxes
- Slipcovers
- Registration cards
- Labels for the disks containing the software product
- Warranties or license agreements, and
- Promotional flyers.

[1] Iban is the largest ethnic group making up about 44% of the population of Sarawak, a state in Malaysia [3]. The language of the Ibans is also known as Iban and the language uses the Roman alphabet.

This paper focuses on the message files making up the user interface spreadsheet. These message files contain the language-dependent components which were extracted from the code of a spreadsheet in an internationalisation exercise. In the internationalisation exercise, the language-dependent components were identified by viewing the user interface of the spreadsheet. These language-dependent components include the menu names, the menu item names, status bar displays of cell contents (text, value, formula), error messages, instructions, commands and dialog messages of the spreadsheet. Next, a thorough search was undertaken to identify language-sensitive components embedded or hidden in the source code. These hidden components included the interactions such as the selection of "Yes" or "No" to a requester. All these language-dependent components were then placed in a message file. Thus, the first message file contained all the interface components in English. Other message files currently available are Mäori and Bahasa Melayu (Malaysia's national language). Appendix 1 shows an example of a message file in English.

As there were a number of message files available, each containing different messages translated to different languages, an algorithm was added to the spreadsheet to allow the use of different message files in the same spreadsheet. Selected message files are loaded into memory depending on which language is chosen via the use of ALT-key combination. A more detailed report of the internationalisation exercise is forthcoming [5]. In the next section, we will describe the translation process using the translation of the message file from English to Iban as an example.

3.0 FRAMEWORK FOR TRANSLATION

The translation of the message file is carried out in a number of steps. Some of the steps include preparing the messages for translation, identifying the translator, conducting the translation and ensuring the messages "fit" the screen real estate. The ensuing sections will describe in detail the framework to carry out the translation.

3.1 Prepare the Message for Translation

Before a message file is translated, the original text must be written in simple and concise English. As with writing for other purposes, the original text has to be written for a specific audience. Uren et al. [4] suggest writing the original text for the international audience. They recommend that a substantial amount of information about the target culture is collected and considered before the original document is passed to the translator. This information may include the target cultures' religion, customs, taboos, beliefs, values, learning styles, presentation styles, depictions of people, animals, sounds and colours. This step is required to ensure that the resulting document will not contain materials that will offend the target audience. Examples of cultural pitfalls can be found in Chapter 5 of Dave Taylor's book [6] and in the article by Russo and Boor [7].

Guidelines for preparing a text for translation have been provided in several publications [1, 4, 8, 9]. These guidelines should not only be used in the writing process but also in the translation process. In our case, the message file in English was checked to ensure the guidelines were followed. As the translation to Bahasa Melayu had already been made [5], the Bahasa Melayu translation was also made available to the translator to take advantage of the similarities between Bahasa Melayu and Iban.

Once the final message file is prepared for translation, an appropriate translator is required.

3.2 Identify the Translator

The translator should, ideally, speak and write the target language fluently. He or she must be fluent in the language of the original document and, preferably, be a current or recent resident of the target country. The translator should also be sensitive to the target cultures' current social concerns.

The translator must be computer literate to ensure that computing related concepts are translated correctly. Computer scientists take for granted simple concepts like "save" to mean save the current file and not in reference to saving money. Barbour and Keegan [10] report that when translating a computer science course manual from English to Mäori, three languages are actually involved in the translation; English, Mäori and the computer language.

An example of an ideal translator is Te Taka Keegan who translated the English message files into Mäori. Te Taka is fluent in Mäori and English, and has about ten years of computing experience.

In many cultures, finding people with the appropriate expertise is difficult. This problem is becoming less of an issue with more people from minority cultures pursuing tertiary level education in computer science.

In the case of the Iban translation, one of the authors (Yeo) acted as a localiser, that is, the person who could communicate with the translator and is also aware of the computing concerns. The localiser was successful in finding an Iban translator at the University of Waikato, Hamilton, New Zealand. The translator, we sought assistance from was Kelvin John. Kelvin is an Iban and speaks fluent Iban, Kelabit (another ethnic group in Sarawak), Bahasa Melayu and English. He is a Masters student studying anthropology at the University of Waikato. Although Kelvin has been in New Zealand for the past five years, he still returns to Sarawak each year. Kelvin has also used computers for more than a year and translated the message file into Iban.

The next step after identifying a translator is to carry out the translation.

3.3 Conduct the Translation

The document should be translated following the guidelines referred to in Section 3.1. In addition, a translation table should be provided. This table contains a glossary of terms used in the translation, that is, the terms in both the original language and in the target language. If such a table does not exist, it should be created during the translation. This table will ensure that the same terms and phrases are used consistently.

In the Iban translation example, we provided Kelvin (the Iban translator) with the English and Bahasa Melayu translation of the messages files. There are some words in both Bahasa Melayu and Iban that share the same meaning, and have similar pronunciation and spelling. We asked Kelvin to write down words that had no equivalent meaning in Iban. He also suggested new words to fill in the gaps. In this way, a record of the translation process was obtained.

For example, the word "spreadsheet" has no equivalent in Iban. If a word does not exist, a word with similar meaning can be used. Kelvin suggested the Iban word *ngancau* which means "something which can be spread". This translation followed the same process of translation as the Bahasa Melayu word *hamparan* which literally means "something that can be spread". Another Iban informant (Justin) agreed with this translation. This agreement is important as different people may use different words for the same meaning. If a standardised terminology does not exist, there will be difficulty in translating future works. In Malaysia, the computer terms and phrases in Bahasa Melayu were created by a committee of computer science experts in collaboration with *Dewan Bahasa dan Pustaka*, the "custodian" of Bahasa Melayu. Similarly, in New Zealand there is a commission which oversees the creation and standardisation of the new words, phrases or computer-related terms in Mäori. To the authors' know-ledge, although there exists an organisation which looks after the Iban language, there is no organisation that deals with computer-related language in Iban. The lack of provision for monitoring computing terms may change with more Iban people graduating with computer science degrees.

In situations where we do not have words close to the original meaning, we can draw metaphors from the language that closely describe the ideas. For example, a ridge in Mäori is *Ripa* from which *te ripanga*, the spreadsheet, is derived (*ripanga* is actually an ordered heap). A better metaphor for the spreadsheet that would be culturally correct is *tukutuku* (from the word *tuku*, catch in a net). The *tukutuku* is a lattice like structure (found in the meeting house which contains the family history), metaphorically, a net for catching ideas. The spreadsheet is thus, culturally, a net for capturing ideas, spaces for ensnaring ideas. It is interesting to note in passing that there is only one academic study into *tukutuku.* This study was cited in a recent anthropological survey into visual imagery [11]. This study provides information about the form of the *tukutuku* but not the content. The content (or family history) is recorded in the *tukutuku* and is women's knowledge, knowledge passed along maternal lines. The lack of knowledge of the *tukutuku* metaphor among men may have been the reason why this appropriate cultural metaphor was not adopted for the spreadsheet.

In the worst case scenario, words from another language are often used where there is no easy cross-cultural translation available. This "borrowing" of words also occurs frequently in English, for example, the use of French words like *rendezvous* or *boutique*. In the Iban language case, words like *cetak* (print) and *perpuluhan* (decimal) were "borrowed" from Bahasa Melayu as no Iban equivalent exists. The approach used in translating or creating new words is summarised in Fig. 1.

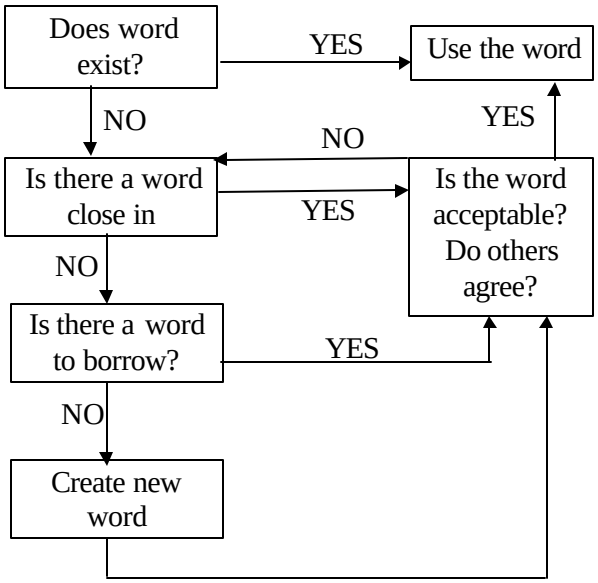

Fig. 1: Translating text and creating words in another language

After the translation has been completed, back translation can be carried out to ensure the translation has been done correctly. This process is carried out by another expert who translates back to the original language. This iteration should ideally be done for all the translated material. Refer to Appendix 1 for the final Iban message file. However, if the translated documentation is huge, a sample of the document can be back translated, say every thousandth paragraph of the document [4].

We must be careful when transliterating words or phrases especially if the translator is not familiar with computing concepts. Russo and Boor [7] gave an example of a transliteration by Sun Microsystems in which the word "menu" was translated to Cantonese as "a list of food

items". Furthermore, we need to know what words can be used depending on the context of the situation. For example, "no" in Iban can be expressed as *anang* (don't do it), *enda* (don't know), *na'iboh* (no need).

We must also be aware that the same language may be spoken in different countries and thus may contain different connotations. In such languages, for example, it may be inappropriate to address an urban accountant in tones typical for a rural farmer or vice versa [1].

When the translation has been completed, the translation is incorporated into the spreadsheet program. Fig. 2 shows the Iban translation fitted into the spreadsheet. Note that the first line of the message overflows to the next line. The problem of fit will be dealt with in the next section.

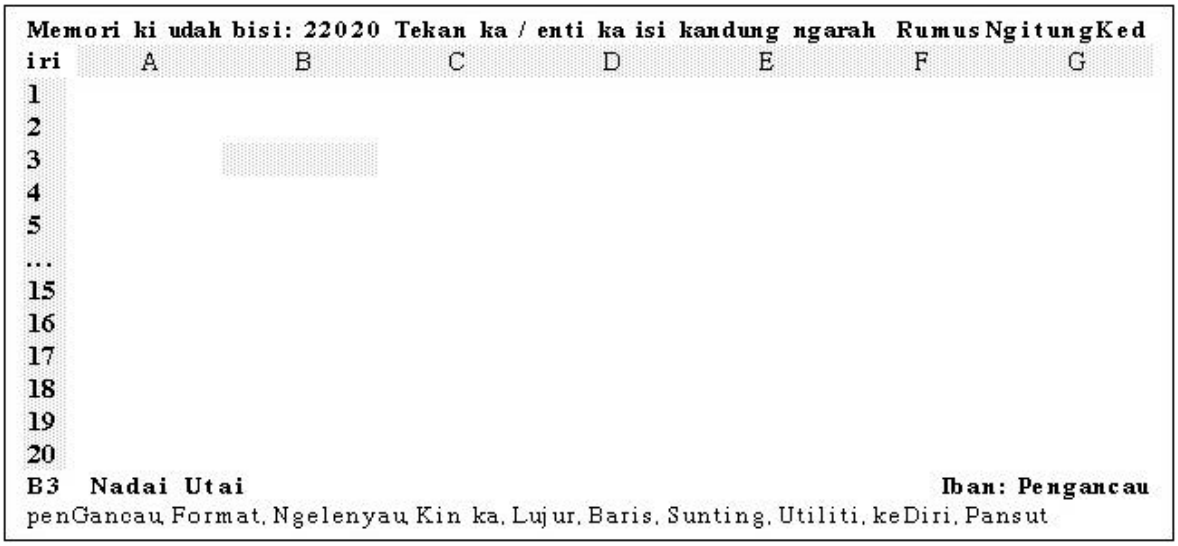

Fig. 2: Screen dump of the spreadsheet after first translation

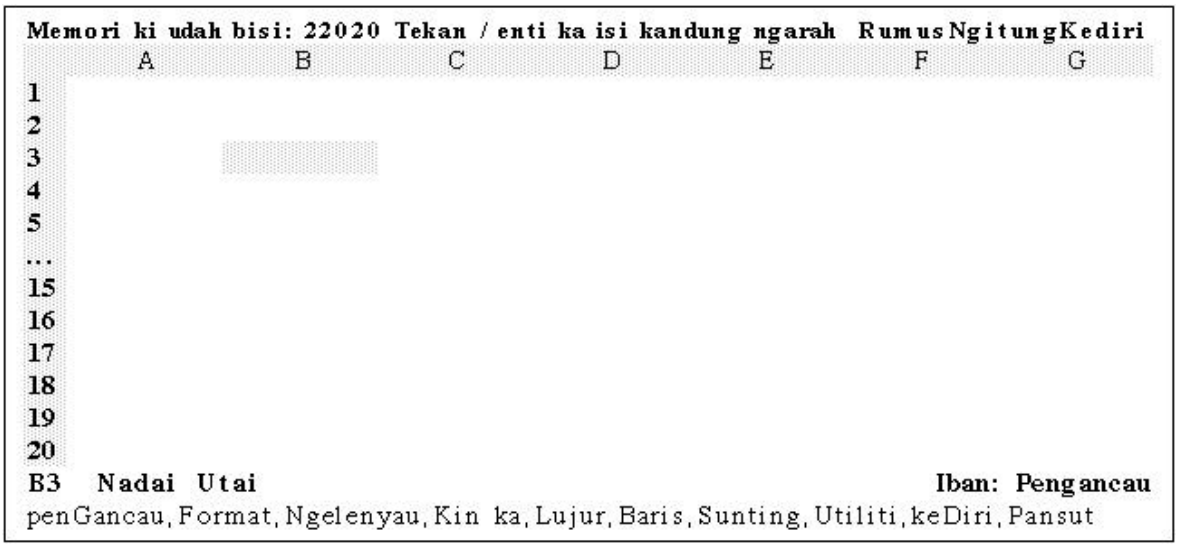

Fig. 3: Screen dump with messages that fit

3.4 Fit Translation into Software

This section describes how the translated components were fitted to the user interface. Since the text did not fit the screen, the translator was consulted to re-translate the text. The localiser and the translator either truncated the words or re-translated the text to words with fewer characters. In an acceptable translation, the final message text must still retain the original meaning.

For example in our case, the message *Tekan ka / enti ka isi kandung ngarah* was reduced to *Tekan / enti ka isi kandung ngarah* so that the first line can fit onto the screen. Compare the first line in Fig. 2 and Fig. 3. The translator removed the word *ka* after *Tekan* while still retaining the original meaning.

Once the translation fits the screen, the target users, both experienced and beginners, should test the new user interface.

3.5 Test Software with Users

Testing is the best approach to ensure the software is acceptable to the target users. The testing process should also include checks on the comprehensibility and usability of items such as the help file and user manuals. The Iban spreadsheet will be tested by fluent speakers in Sarawak. The target group will probably be first year university students who have yet to learn about spreadsheets but are fluent in Iban. The thinking-aloud method, described as one of the most important methods for user interface evaluation [12], can be used in this case. The think-aloud technique requires the subjects to "think out loud" while completing the tasks prescribed to them. By analysing the subjects' thoughts, we would be able to isolate usability and comprehension problems that exist in the translated spreadsheet.

4.0 SUMMARY

In the Introduction section, we gave some reasons why there should be software for minority languages. As the focus of this article is on translation, we translated the English messages of a spreadsheet to Iban. The translation process we went through is summarised in Fig. 4. Some translation problems that arose during the translation process were discussed. For example, certain words in English have no equivalent in Iban. We gave examples on how to create new words for non-existing words. After the message translation was incorporated in the spreadsheet, the messages did not fit on the screen. We described how this problem was fixed by the translator and localiser.

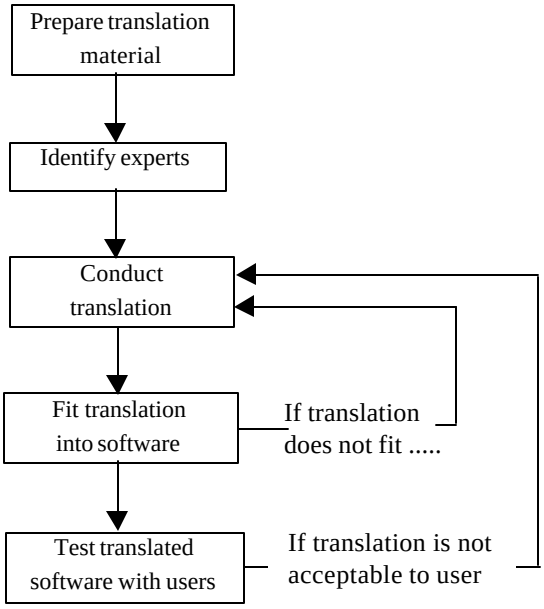

Fig. 4: Translating the message files for the software

5.0 FURTHER WORK

The Iban translation we conducted is not complete. We need to show the translation to Iban language experts and the potential users to obtain their opinions of the translation. One possible source of Iban language experts is the Education Department in Sarawak. These experts would know the standard Iban used since they would be in charge of the Iban language syllabus as well as authoring Iban textbooks.

Also, the software, with the translated Iban messages, needs testing with the target groups. Further work could be carried out to investigate whether Iban students do learn faster if provided with a software in their native tongue.

Our translator has observed that there are regional dialects of Iban. Thus, Ibans from different areas might require different Iban translations of the same spreadsheet. Our strategies for localising software can easily accommodate for changes in the message files. There is no reason in principle why our localisation process could not be extended to GUI based spreadsheets when software houses release their source code for research purposes. People wishing to test the software should contact the authors at their institutional address.

ACKNOWLEDGMENTS

The authors would like to thank Kelvin John and Justin for their assistance in carrying out the Iban translation. Borland International is thanked for their release of their source code which facilitated this research. The authors would also like to thank the referees of the Malaysian Journal of Computer Science for their critique on an earlier version of this paper.

REFERENCES

[1] Apple, *Guide to Macintosh Software Localization*. Addison Wesley, Reading, Mass., 1992.

[2] D. Griffiths, S. Heppell, R. Millwood and G. Mladenova,"Translating Software: What It Means and What It Costs for Small Cultures and Large Cultures". *Computers in Education*, Vol. 22, No. 1/2, 1994, pp. 9-17.

[3] Berita Publishing Group, *Information Malaysia 1994*. Berita Publishing Group, 1994.

[4] E. Uren, R. Howard and T. Preinotti, *Software Internationalization and Localization: An Introduction*. Van Nostrand Reinhold, New York, 1993.

[5] R. Barbour and A. Yeo. "Internationalising a Spreadsheet for Pacific Languages". *New Zealand Journal of Computing*, Dec. 1996.

[6] D. Taylor, *Global Software: Developing Applications for the International Market.* Springer-Verlag, New York. 1992.

[7] P. Russo and S. Boor. "How Fluent is your Interface? Designing for International Users", in *Proceedings of INTERCHI '93 Conference on Human Factors in Computing Systems: INTERACT '93 and CHI'93, ACM Amsterdam.* April 1993, pp. 342-347.

[8] N. Kano, *Developing International Software for Windows 95 and Windows NT*. Microsoft Press, Redmond, Washington, 1995.

[9] S. O'Donnell, *Programming for the Whole World: A Guide to Internationalization*, Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[10] R. Barbour and T. Keegan , "Education in Technology for the LCTLs: Nga Tautono Rorohiko", in *Proceedings of NFLRC Symposium, University of Hawai'i, http://www.lll.hawaii.edu/nflrc/NetWorks/NW2/morn.html*, 8-12 July 1996.

[11] Neich Roger, *Painted Histories*. Auckland University Press, Auckland, 1993.

[12] J. Nielsen, "Evaluating the Thinking-Aloud Technique for Use by Computer Scientists". In *H. Hartson and D. Hix (Eds.). Advances in Human-Computer Interaction.* Vol. 3, pp. 69-82.

BIOGRAPHY

Alvin Yeo is a tutor currently on study-leave from the Faculty of Information Technology, Universiti Malaysia Sarawak to complete his Ph. D. (Computer Science) at the University of Waikato. His research interests include software internationalisation and localisation, and culture's effects on human computer interaction. He is a student member of ACM and SIGCHI.

Dr. Robert Barbour is a senior lecturer in a Post-graduate Centre for Research in Computer Science, Science, Mathematics and Technology Education Research, University of Waikato, Hamilton, New Zealand. He has a background in Social Sciences, Education and Computer Science. He is interested in researching software tools which preserve human rights while promoting international communication. He is a member of the New Zealand Royal Society and is involved with International Standards work.

APPENDIX 1

	English	Iban
1	"English: Spreadsheet"	"Iban: Pengancau"
2	"Press E to select English"	"Tekan ka I enti ka jaku Iban""
3	"Can't open the file"	"Fail endah ulih dibuka"
4	"Press / for the list of commands"	"Tekan ka / enti ka isi kandung ngarah"
5	"Memory Available:"	"Memori ki udah bisi"
6	"ERROR"	"Penyalah"
7	"Not enough memory to allocate cell"	"Enda cukup memori ke ngalai ke sel"
8	"Empty"	"Nadai utai"
9	"Text"	"Tulis"
10	"Value"	"Ungkus ki bërega"
11	"Formula"	"Rumus"
12	"AutoCalc"	"NgitungKediri"
13	"Formula"	"Rumus"
14	"Enter the file name of the spreadsheet:"	"Pasuk ke nama fail"
15	"Press any key to continue"	"Tekan aja enti ka nyambung ngancau"
16	"Enter the new column width:"	"Pasuk ka pemesai lujur baru"
17	"The file exists. Do you want to overwrite it?"	"Fail to bisi. Ka nuan nganti fail ka lama?"
18	"Not enough memory for entire spreadsheet"	"Enda cukup memori kena pengancau"
19	"That is not a FIRST spreadsheet"	"Nya ukai fail pengancau FIRST"
20	"The file does not exist"	"Nadai fail ki bakai tu"
21	"Enter the cell to go to:"	"Tama sel kä dë kä :"
22	"You must enter a number from %d to %d."	"Nuan patut nama ka numbur ari %d ka %d"
23	"That is not a legal cell"	"Enda betul sel nya"
24	"Enter the first cell to format:"	"Tama ka sel ki terubah enti ka format:"
25	"Enter the last cell to format:"	"Tama ka sel ki penghabis enti ka format:"
26	"The row or the column must be the same"	"Baris atu lujur mesti ka sama"
27	"Do you want the cell right-justified?"	"Ka nuan ngasoh sel nya lurus mayang ba sepia kanan?"
28	"Do you want numbers in a dollar format?"	"Ka nuan ngaso numbur nya dalam format RM?"
29	"Do you want commas in numbers?"	"Ka nuan ngasoh numbur nya bisi koma?"
30	"How many decimal places should the number be rounded to?"	"Berapa alai perpuluhan ti perlu dibundarkan ka tiap numbur?"
31	"Do you want to print in 132 columns?"	"Ka nuan cetak 132 lujur?"
32	"Enter the file name to print to, or press ENTER to print on the printer"	"Tama ka nama fail ka dicetak alam atau tekan ka ENTER enti ka cetak?"
33	"Print the border?"	"Cetak ka sempadan?"
34	"Loading..."	"Alam proses muat ka fail ..."
35	"Saving..."	"Alam proses nyimpan ka fail..."
36	"Save current spreadsheet?"	"Ka nuan nyimpan pengancau tu?"
37	"Parser stack overflow."	"Tindanan penghurai melimpah atas"
38	"Spreadsheet, Format, Delete, Goto, Col, Row, Edit, Utility, Auto, Quit"	"penGancau,Format,Ngelenyau,Kin ka,Lujur,Baris,Sunting,Utiliti,keDiri,Pansut"
39	"SFDGCREUAQ"	"GFNKLBSUDP"
40	"Load, Save, Print, Clear"	"Padat ka, nYimpan, Cetak ka, Ngelenyau ka"
41	"LSPC"	"PYCN"
42	"Insert, Delete, Width"	"Tama ka, Ngelenyau ka, peMesai"
43	"IDW"	"TNM"
44	"Insert, Delete"	"Tama ka, Ngelenyau ka"
45	"ID"	"TN"
46	"Recalc, Formula display"	"Itung baru, Padah ka rumus"
47	"RF"	"IP"
48	"YN"	"AN"
49	"$"	"RM"