Working Paper Series
ISSN 1170-487X

Localising A Spreadsheet: An Iban Example

by Alvin Yeo and Robert Barbour

Working Paper 97/18
July 1997

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Localising A Spreadsheet: An Iban Example

Alvin Yeo
Department of Computer Science
University of Waikato, Hamilton, New Zealand.
Email: awy@cs.waikato.ac.nz

Robert Barbour
Science, Mathematics and Technology Education Research Centre
University of Waikato, Hamilton, New Zealand
Email: r.barbour@waikato.ac.nz

ABSTRACT

Presently, there is little localisation of software to smaller cultures if it is not economically viable. We believe software should also be localised to the languages of small cultures in order to sustain and preserve these small cultures. As an example, we localised a spreadsheet from English to Iban. The process in which we carried out the localisation 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.

Introduction

Most commercial software production is dominated by products from computer firms in the United States (US). For example, the Apple Macintosh operating system is available in 30 major languages [1]. This number of languages would ensure that the localisation of that system is economically viable. Languages with relatively fewer speakers, like Mäori or Iban, 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 view point, holding rather that having access to ones 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. in [5] 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 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 Iban text interface. 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.

The Iban language is introduced in the next section. Items requiring translation are outlined in Section 3. Possible approaches for translating various components of the software are described in Section 4.

The Iban Language

Iban is spoken by the Iban ethnic group in Sarawak, a state in Malaysia. There are about half a million Iban speakers. The Iban language uses the Roman alphabet. Missionaries were probably the first to record Iban in the written form.

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, for example, text strings for menus, dialog boxes, help files and error messages.

Uren et al. (1994) [3] 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.

This paper focuses on the message files of the software component. These message files contain the language-dependent components which were extracted from a spreadsheet in an internationalisation exercise. 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 in a particular language. A number of message files are available, each containing different messages translated to different languages. Selected message files are loaded into memory depending on which language is chosen. Appendix 1 shows an example of a message file in English. The process of extracting the language-dependent components from the spreadsheet is documented in [4].

In the next section, we will describe the translation process. We will use the translation of the message files from English to Iban as an example.

Framework For Translation

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. [11] suggest writing the original text for the international audience. He recommends that as much information about the target cultures 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 [10] and in the article by Russo and Boor [9].

Guidelines for preparing a text for translation are found in [1,6,8,11]. 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 (Malaysia's national language) had already been made in [4], 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 complete, an appropriate translator is required.

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' 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 [3] 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 message files into Mäori. Te Taka is fluent in Mäori and English, and has about ten years of computing experience.

In many cultures, problems may exist in looking for people with the appropriate expertise. 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 acted as a localiser, that is, the person who could communicate with the translator and who is also aware of the computing concerns. The localiser was successful in finding an Iban translator at the University of Waikato, New Zealand. The translator we sought assistance from was Kelvin John. Kelvin is an Iban and speaks fluent Iban, Kelabit (an 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.

After we identified the translator, we will next, translate the message file.

Carry Out Translation

The document should be translated following the guidelines referred to in Section 4.1. 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 meaning 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 work. 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. 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 computer science graduates.

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 *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 (Hinekahukura Aranui, pers. com.) [2]. 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 [7]. 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. 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 or identified. This "borrowing" of words also occurs 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 was found. The approach used in translating or creating new words is summarised in Figure 1.

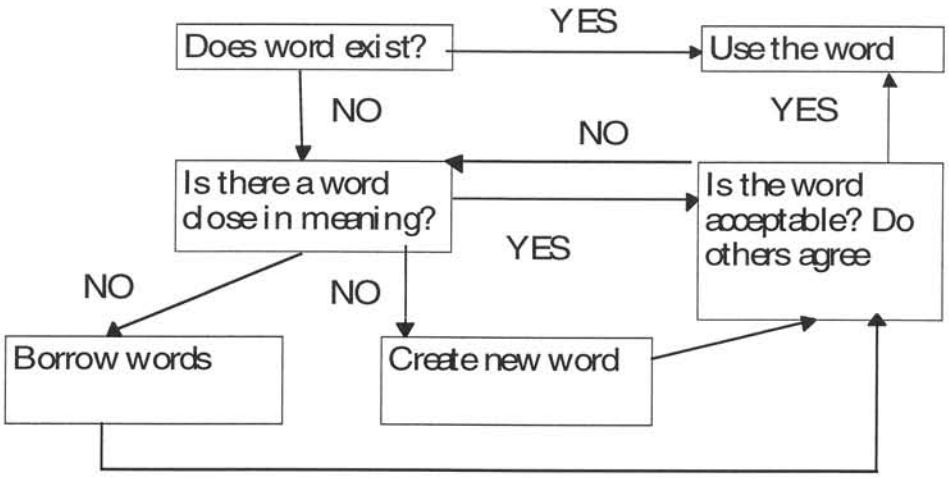

Figure 1: Translating text and creating words in another Language

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 2 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 [11].

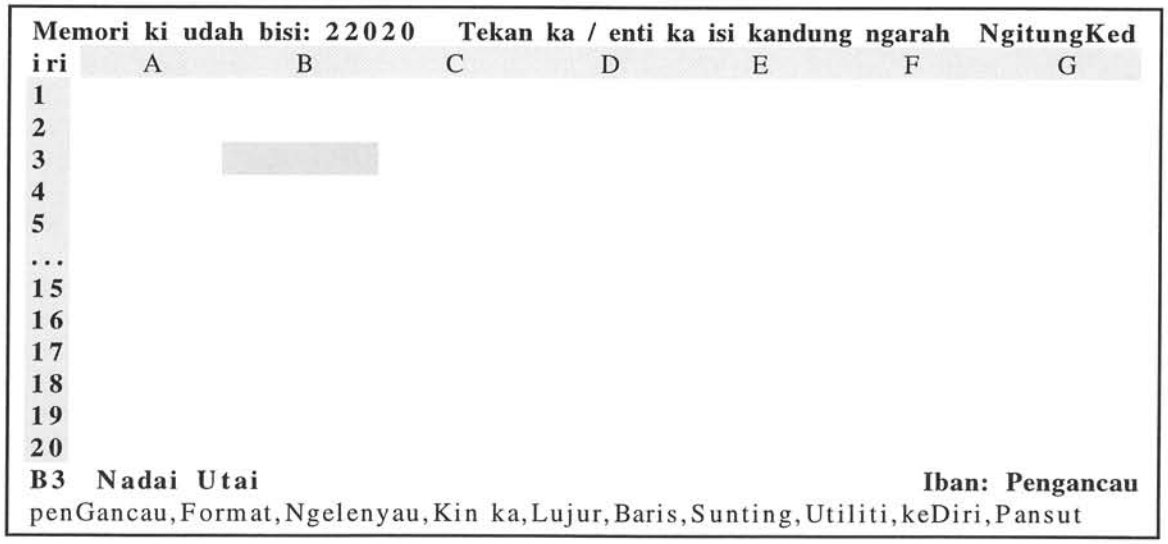

Figure 2: Screen dump of the spreadsheet after first translation

We must be careful when transliterating words or phrases especially if the translator is not familiar with computing concepts. Russo and Boor [9] 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 intonations and connotations in the different regions. 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 software. Figure 2 below shows the Iban translation fitted into the spreadsheet. Note that the first line of the messages overflows to the next line. The problem of fit will be dealt with in the next section.

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 re-translated the text to words with fewer characters or truncated the words. 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 Figure 2 and Figure 3. The translator removed the word *ka* after *Tekan* while still retaining the original meaning.

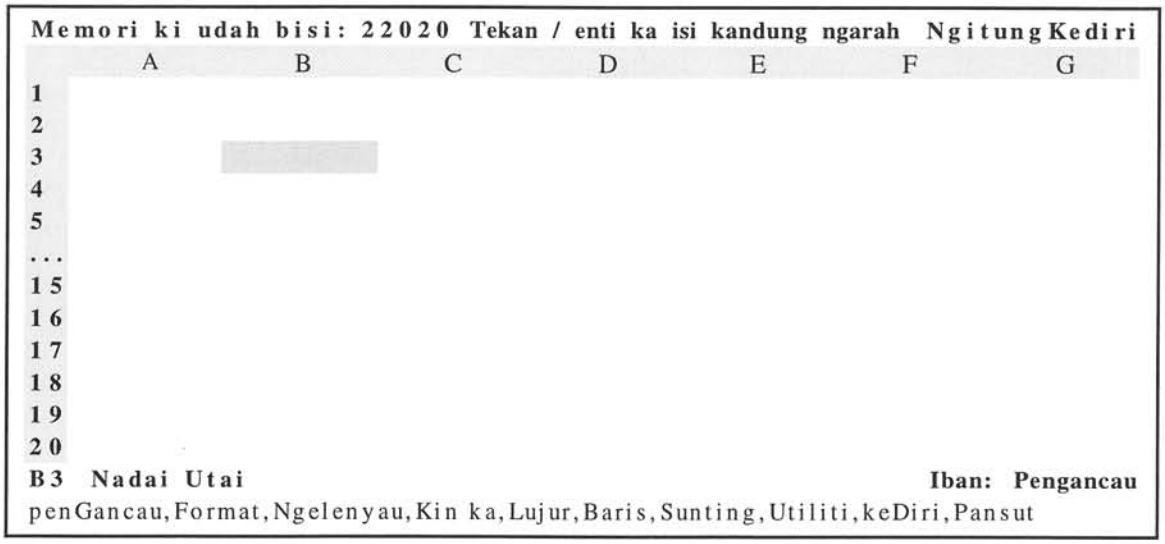

Figure 3: Screen dump with messages that fit.

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

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.

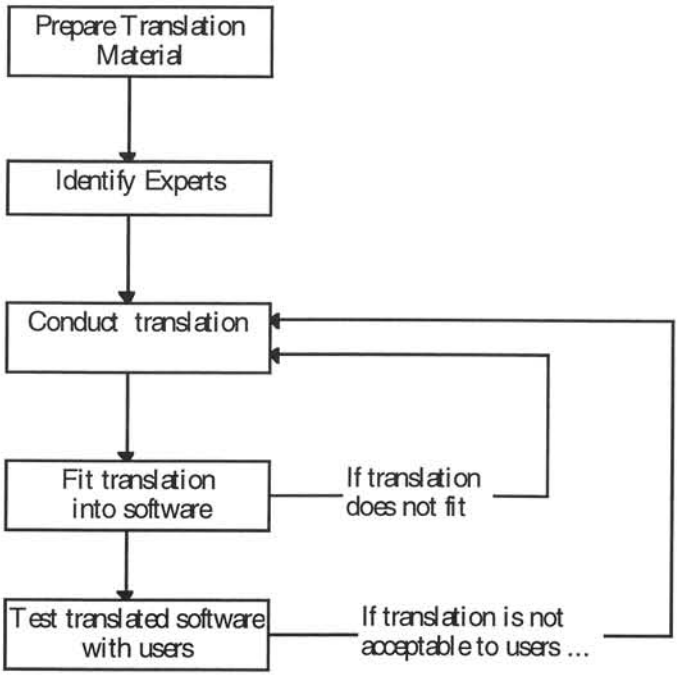

Figure 4: Translating the message files for the software

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 Figure 4. Some translation problems that arose during the translation process were discussed. For example, certain words in English have no equivalent words 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 well in the screen. We described how this problem was fixed by the translator and localiser.

Further Work

The Iban translation we conducted is not complete. We need to show the translation to the potential users and Iban language experts to obtain their opinion of the translation. The software, with the translated Iban messages, needs testing with the target groups.

Our translator has observed that Iban is spoken differently in different parts of Sarawak. 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. 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.

References

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

[2] Aranui T 1996 Personal Communication,

[3] Barbour, R. and Te Keegan, (1996) : Education in Technology for the LCTLs: Nga Tautono Rorohiko. NFLRC Symposium (University of Hawai'i, 8-12 July). http://www.lll.hawaii.edu/nflrc/NetWorks/NW2/morn.html

[4] Barbour, R. and Yeo, A. (1996). Internationalising a Spreadsheet for Pacific Languages.

[5] Griffiths, D., Heppell, S., Millwood, R. and Mladenova, G. (1994) : Translating Software: What It Means and What It Costs for Small Cultures and Large Cultures, *Computers in Education*, **22**(1/2), pp9-17.

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

[7] . Nei ch Roger 1993 Painted Histories. Auckland University Press, Auckland.

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

[9] Russo, P. and Boor, S. (1993) : *How Fluent is your Interface? Designing for International Users,* INTERCHI '93 Conference on Human Factors in Computing Systems: INTERACT '93 and CHI'93, (Amsterdam, 24-29 April), ACM Press. pp342-347.

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

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

Appendix 1

```
1       "English: Spreadsheet"
2       "Press E to select English"
3       "Can't open the file"
4       "Press / for the list of commands"
5       "Memory Available:"
6       "ERROR"
7       "Not enough memory to allocate cell"
8       "Empty"
9       "Text"
10      "Value"
11      "Formula"
12      "AutoCalc"
13      "Formula"
14      "Enter the file name of the spreadsheet:"
15      "Press any key to continue"
16      "Enter the new column width:"
17      "The file exists.  Do you want to overwrite it?"
18      "Not enough memory for entire spreadsheet"
19      "That is not a FIRST spreadsheet"
20      "The file does not exist"
21      "Enter the cell to go to:"
22      "You must enter a number from %d to %d."
23      "That is not a legal cell"
24      "Enter the first cell to format:"
25      "Enter the last cell to format:"
26      "The row or the column must be the same"
27      "Do you want the cell right-justified?"
28      "Do you want numbers in a dollar format?"
29      "Do you want commas in numbers?"
30      "How many decimal places should the number be rounded to?"
31      "Do you want to print in 132 columns?"
32      "Enter the file name to print to, or press ENTER to print on the printer"
33      "Print the border?"
34      "Loading..."
35      "Saving..."
36      "Save current spreadsheet?"
37      "Parser stack overflow."
38      "Spreadsheet, Format, Delete, Goto, Col, Row, Edit, Utility, Auto, Quit"
39      "SFDGCREUAQ"
40      "Load, Save, Print, Clear"
41      "LSPC"
42      "Insert, Delete, Width"
43      "IDW"
44      "Insert, Delete"
45      "ID"
46      "Recalc, Formula display"
47      "RF"
48      "YN"
49      "$"
```

Appendix 2

```
1     "Iban: Pengancau"
2     "Tekan ka I enti ka jaku Iban""
3     "Fail endah ulih dibuka"
4     "Tekan ka / enti ka isi kandung ngarah"
5     "Memori ki udah bisi"
6     "Penyalah"
7     "Enda cukup memori ke ngalai ke sel"
8     "Nadai utai"
9     "Tulis"
10    "Ungkus ki bërega"
11    "Rumus"
12    "NgitungKediri"
13    "Rumus"
14    "Pasuk ke nama fail"
15    "Tekan aja enti ka nyambung ngancau"
16    "Pasuk ka pemesai lujur baru"
17    "Fail to bisi. Ka nuan nganti fail ka lama?"
18    "Enda cukup memori kena pengancau"
19    "Nya ukai fail pengancau FIRST"
20    "Nadai fail ki bakai tu"
21    "Tama sel kä dë kä :"
22    "Nuan patut nama ka numbur ari %d ka %d"
23    "Enda betul sel nya"
24    "Tama ka sel ki terubah enti ka format:"
25    "Tama ka sel ki penghabis enti ka format:"
26    "Baris atu lujur mesti ka sama"
27    "Ka nuan ngasoh sel nya lurus mayang ba sepia kanan?"
28    "Ka nuan ngaso numbur nya dalam format RM?"
29    "Ka nuan ngasoh numbur nya bisi koma?"
30    "Berapa alai perpuluhan ti perlu dibundarkan ka tiap numbur?"
31    "Ka nuan cetak 132 lujur?"
32    "Tama ka nama fail ka dicetak alam atau tekan ka ENTER enti ka cetak?"
33    "Cetak ka sempadan?"
34    "Alam proses muat ka fail ..."
35    "Alam proses nyimpan ka fail..."
36    "Ka nuan nyimpan pengancau tu?"
37    "Tindanan penghurai melimpah atas"
38    "penGancau,Format,Ngelenyau,Kin ka,Lujur,Baris,Sunting,Utiliti,keDiri,Pansut"
39    "GFNKLBSUDP"
40    "Padat ka, nYimpan, Cetak ka, Ngelenyau ka"
41    "PYCN"
42    "Tama ka, Ngelenyau ka, peMesai"
43    "TNM"
44    "Tama ka, Ngelenyau ka"
45    "TN"
46    "Itung baru, Padah ka rumus"
47    "IP"
48    "AN"
49    "RM"
```