# Internationalisation – the user perspective

\\stephen\c\mpe\projects\international\internationalisation.v2.doc
Revised 13 August 1998

**Authors**:
Stephen Pelc, MicroProcessor Engineering
Willem Botha, Construction Computer Software
Nick Nelson, Micross Electronics
Peter Knaggs, Bournemouth University

Contact:
Stephen Pelc
MicroProcessor Engineering
133 Hill Lane
Southampton SO15 5AF
England

Tel: +44 1703 631441
Fax: +44 1703 339691
Net: sfp@mpeltd.demon.co.uk
Web: http://www.mpeltd.demon.co.uk

## Introduction

Internationalisation (or localisation) is becoming a major issue for several of MPE's clients. These clients are of the opinion that the current ANS definitions of character related words are broken for multi-language use. In addition, in a world of 16 bit and multibyte character sets, the standard appears confused between characters and bytes.

Construction Computer Software (CCS) have 15 years experience of supporting multiple languages with Forth. Their software is used all over the world, and in languages for which Unicode support is not yet available. Coping with multibyte character sets must be included within an internationalisation system.

Micross Electronics have ported applications to run in Japanese and several European languages.

The word set presented here is based on a system that has been in use at CCS for many years.

## Objectives

A modification of the CCS word set has been designed. The authors are prepared to propose as the basis of an ANS standard. This is referred to as the LOCALE word set. The basis of this is that all strings for internationalisation will be compiled as LOCALE structures, and all access to the strings is through these structures. It appears that the following word set is adequate in the first place. The word set is designed to cope with character sets that are of different size to the native set. For example, the development language may be 8 bit ASCII, whereas the language strings may be returned in Unicode (16 bit) or UTF8 or other multibyte character sets. Note that use of the standard Forth words to navigate through strings under these conditions is impossible.

The word set is split into user and implementer groups to indicate what factors need to be language sensitive. It is also likely that all LOCALE structures will need to be linked in case reindexing of hash tables or other internal structures is necessary.

The word **L"** is proposed for language sensitive strings, and behaves in a similar way to the ANS word **C"**, but returns a string identified known as a LOCALESTRUCT from which the required language string can be extracted. The reason for this is so that text information in the native development language is still available in the source, making source maintenance much easier because the intention of the string is still available to the developer. In addition, the Forth compiler can be extended to produce a text file containing the native strings.

In practice, many applications not are localised by the software developer, but their agents in other countries. The use of **L"** permits the software developer to produce tools that will produce text files that can be edited and converted to another language locally without dependency on computer language or operating system specific tools such as resource compilers and managers.

This document is not an ANS proposal yet, but is a working paper from which we will generate an ANS proposal.

# The LOCALE word sets

## *User LOCALE word set*

**LANGUAGE** \ lang -- ; lang is a language code, e.g. Windows constant
Sets the current language for the LOCALE system. Note that several variants of a basic language may exist, e.g. Portugese and Brazilian dialects of Portugese.

**GET-LANGUAGE** \ -- lang
Returns the language code last set by **LANGUAGE**.

**COUNTRY** \ country -- ; country is a country code, e.g. Windows constant
Sets the current country for the LOCALE system. This is necessary for countries that have several languages, e.g. Belgium, Canada, Switzerland, South Africa, Russia, China.

**GET-COUNTRY** \ -- country
Returns the country code last set by **COUNTRY**.

**L"** \ -- ; -- localestruct ; L" <native text>"
Compiles a LOCALE structure with an inline string compiled as a counted string in the native language of the development system. At run time, the address of the LOCALE structure is returned. Other words use this structure to extract language specific information.

**(L")** \ -- localestruct
Steps over an inline LOCALE structure and returns its address

**LOCALE-FETCH** \ localestruct -- $addr
Fetches a string in the current language that corresponds to the native string compiled with the LOCALE structure, and returns its address. The format of the string at $addr is implementation dependent, and may depend on the operating system and character encoding in use.

**LOCALE-COUNT** \ $addr – addr len(au)
Given the address of a translated language string returns the address of the first character, and its length in address units. This function enables translated strings to be copied by words such as CMOVE.

### *Implementer word set*

This word set need not be part of the standard, but is provided here to indicate what is needed in a practical implementation, and to show what is system dependent.

**LOCALE-INDEX** \ localestruct –
Updates the internal data structure. Useful if structures are added and changes to internal structures are required.

**LOCALE-LINK** \ localestruct1 – localestruct2
Given the address of one LOCALE structure, returns the address of the next.

**LOCALE-TYPE** \ addr len(au) –
Displays the LOCALE string whose address and length are given.

**LOCALE$** \ localestruct – c-addr
Given a LOCALE structure, returns the address of the corresponding native string that was compiled by L".

## Impact on ANS Forths

Any word that depends on the current definition of **COUNT** cannot be used for internationalisation. This includes, but is not limited to:

| | | | | |
|---|---|---|---|---|
| COUNT | C@ | CMOVE | /STRING | SEARCH |
| WORD | C! | CMOVE> | PARSE | COMPARE |
| SKIP | SCAN | | | |

Unless all these words are deferred, their use is restricted to the character size of the native Forth. For example most Forth systems are written for a seven or eight bit character set, but the applications written in those Forth systems may have to display messages in 16 bit or multibyte character sets.

It has been proposed that **COUNT** should return the number of address units (ANSism, but normally bytes) used in the string and not the number of characters. In this way, most if not all, words based around **COUNT** will continue to work. Perhaps a word such as **CHARLEN** ( addr -- n ) could count the number of (printable?) characters in the string.

If the words above are to be made language sensitive, then the following others are needed as well.

| | | | | |
|---|---|---|---|---|
| C@++ | C!++ | TOUPPER | TOLOWER | CONCAT |

Since character strings may have multibyte character sets, **C@** and **C!** cannot be used to handle bytes if they are character related. The majority of communication protocols, eg. TCP/IP, are 8 bit byte (octet) oriented, and access to bytes (and other fixed size units) will also be required for Forth. This topic is not covered in this document. Consequently the following words are needed:

| | | | |
|---|---|---|---|
| B@ | B! | BMOVE | BMOVE> |

An alternative approach is recognise that the Cxxx words are really byte operators or operators on at least 8 bits (in cell addressed or bit addressed CPUs), and to define a new set of operators to act on character strings.

## Formatting Considerations

We have not yet discussed the various formatting problems: Date/Time, Calendar, Currency and Number. I will accept that these are based on the current country setting rather than the current language setting. For example English is used in at least nine different countries, each with their own formatting considerations. There are also a few countries that have multiple languages; Canada for example uses both French and English. This was also the main thrust of Peter Knaggs EuroFORTH '97 paper, although that was not complete. I refer you to Chapter 5 of Kano for further discussion.

## Relevant Standards

There are a number of different standards that can be used for country and language codes. The codes used by Microsoft (See Appendix K of Kano) are recommended for two reasons:

1) They are the most complete set we have thus far seen.
2) Most people working in the multi-lingual environment will have already come across them.

There are also a few problems with using these codes:

1) They do not conform to any standard (or standard that we know of), although being used by Microsoft they are a de-facto standard.
2) The "primary language" ID is well defined, however, the "secondary" (or country) ID is not so well defined. A combination of both the "primary" (or language) ID and the "secondary" (or variant/country) ID is required to fully identify a given locale. For example, Britain requires a primary ID of 9 (English) and a secondary ID of 8, giving a country code of 0809 (Hex), however, the same secondary code, combined with the primary code 1 indicates "Iraq".

[ISO90] International Standards Organisation. *Programming Languages—C*. ISO/IEC 9899:1990.

[ISO95] International Standards Organisation. *Information Technology—Programming Languages—Ada*. ISO/IEC 8652:1995.

[ISO97] International Standards Organisation. *Information Technology—Programming Languages—Forth*. ISO/IEC 15145:1997.

[ISO98a] International Standards Organisation. *Information technology—Framework for internationalization*. ISO/IEC TR 11017:1998.

[ISO98b] International Standards Organisation. *Information technology—International string ordering—Method for comparing character strings and description of a default tailorable Ordering*. ISO/IEC DIS 14651. 1998.

[ISO98c] International Standards Organisation. *Functionality for internationalization—Specification of cultural conventions*. ISO/IEC DIS 14652. 1998.

## Bibliography

[Kan95] Kano, Nadine. *Developing International Software*. Microsoft Press, 1995.

[Kay94] Kay, Russell. "Software Goes Global." *Byte Magazine* (June 1994): 90–91.

[Kna97] Peter Knaggs. "A Truly International Standard." *Proceedings of the EuroFORTH '97 Conference*, Southampton: MPE Ltd., 1997.

[Kna98] Peter Knaggs (Ed). *International Considerations for the Forth Programming Language, Draft Revision 2*. Bournemouth University, 1998. http://dec.bournemouth.ac.uk/forth/international.

[Nak94] Nakakoji, Kumiyo. "Crossing the Cultural Boundary." *Byte Magazine* (June 1994): 107–9.

[Odo94] O'Donnell, Sandra Martin. *Programming for the World: How to Modify Software to Meet the Needs of the Global Market.* Prentice Hall, 1994.

[Tay92] Taylor, Dave. *Global Software: Developing Applications for the International Market.* New York: Springer-Verlag, 1992.

[UHP93] Uren, Emmanuel, Robert Howard, and Tiziana Perinotti. *Software Internationalization and Localization: An Introduction.* New York: Van Nostrand Reinhold, 1993.

# Microsoft Locale ID codes

| Primary ID | Language | Secondary ID | Country |
|---|---|---|---|
| 00 | Neutral | | |
| 01 | Arabic | 04 | Saudi Arabia |
| | | 08 | Iraq |
| | | 0C | Egypt |
| | | 10 | Libya |
| | | 14 | Algeria |
| | | 18 | Morocco |
| | | 1C | Tunisia |
| | | 20 | Oman |
| | | 24 | Yemen |
| | | 28 | Syria |
| | | 2C | Jordan |
| | | 30 | Lebanon |
| | | 34 | Kuwait |
| | | 38 | United Arab Emirates |
| | | 3C | Bahrain |
| | | 40 | Qatar |
| 02 | Bulgarian | 04 | |
| 03 | Catalan | 04 | |
| 04 | Chinese | 04 | Taiwan |
| | | 08 | People's Republic of China |
| | | 0C | Hong Kong |
| | | 10 | Singapore |
| 05 | Czech | 04 | |
| 06 | Danish | 04 | |
| 07 | German | 04 | Standard |
| | | 08 | Switzerland |
| | | 0C | Austria |
| | | 10 | Luxembourg |
| | | 14 | Liechtenstein |
| 08 | Greek | 04 | |
| 09 | English | 04 | United States |
| | | 08 | Britain |
| | | 0C | Australia |
| | | 10 | Canada |
| | | 14 | New Zealand |
| | | 18 | Ireland |
| | | 1C | South Africa |
| | | 20 | Jamaica |
| | | 24 | Carribean |
| 0A | Spanish | 04 | Traditional Sort |
| | | 08 | Mexican |
| | | 0C | Modern Sort |
| | | 10 | Guatemala |
| | | 14 | Costa Rica |
| | | 18 | Panama |

| | | | |
|---|---|---|---|
| | | 1C | Dominican Republic |
| | | 20 | Venezuela |
| | | 24 | Colombia |
| | | 28 | Peru |
| | | 2C | Argentina |
| | | 30 | Ecuador |
| | | 34 | Chile |
| | | 38 | Uruguay |
| | | 3C | Paraguay |
| | | 40 | Bolivia |
| 0B | Finnish | 04 | |
| 0C | French | 04 | Standard |
| | | 08 | Belgium |
| | | 0C | Canada |
| | | 10 | Switzerland |
| | | 14 | Luxembourg |
| 0D | Hebrew | 04 | |
| 0E | Hungarian | 04 | |
| 0F | Icelandic | 04 | |
| 10 | Italian | 04 | Standard |
| | | 08 | Switzerland |
| 11 | Japanese | 04 | |
| 12 | Korean | 04 | Standard |
| | | 08 | Johab |
| 13 | Dutch | 04 | Standard |
| | | 08 | Belgium |
| 14 | Norwegian | 04 | Bokmål |
| | | 08 | Nynorsk |
| 15 | Polish | 04 | |
| 16 | Portuguese | 04 | Brazilian |
| | | 08 | Standard |
| 17 | Rhaeto-Romanic | 04 | |
| 18 | Romanian | 04 | Standard |
| | | 08 | Moldavia |
| 19 | Russian | 04 | Standard |
| | | 08 | Moldavia |
| 1A | Serbo-Croatian | 04 | Croatan |
| | | 08 | Serbian |
| 1B | Slovak | 04 | |
| 1C | Albanian | 04 | |
| 1D | Swedish | 04 | |
| 1E | Thai | 04 | |
| 1F | Turkish | 04 | |

| | | | |
|---|---|---|---|
| 20 | Urdu | 04 | |
| 21 | Indonesian | 04 | |
| 22 | Ukrainian | 04 | |
| 23 | Byelorussian | 04 | |
| 24 | Slovenian | 04 | |
| 25 | Estonian | 04 | |
| 26 | Latvian | 04 | |
| 27 | Lithuanian | 04 | |
| 28 | Maori | | |
| 29 | Farsi | 04 | |
| 2A | Vietnamese | | |
| 2B | Laotian | | |
| 2C | Kampuchean | | |
| 2D | Basque | 04 | |
| 2E | Sorbian | 04 | |
| 2F | Macedonian | 04 | |

| | | | |
|---|---|---|---|
| 30 | Sutu | 04 | |
| 31 | Tsonga | 04 | |
| 32 | Tswana | 04 | |
| 33 | Venda | 04 | |
| 34 | Xhosa | 04 | |
| 35 | Zulu | 04 | |
| 36 | Afrikaans | 04 | |
| 37 | | | |
| 38 | Faeroese | 04 | |
| 39 | Hindi | 04 | |
| 3A | Maltese | 04 | |
| 3B | Sami | 04 | |
| 3C | Scots Gaelic | 04 | Gaidhilge |
| 3D | | | |
| 3E | | | |
| 3F | | | |

# Contributers' comments

### *Willem Botha*

I have a problem with the user locale word set to indicate if the required string is counted or zero terminated. I propose that locale strings by default have a count for dictionary navigation, and a zero terminator for API calls. The user word set can be extended to give access to the counted string address or the first character of the zero terminated string. The count will exclude the null terminator and the dictionary navigation words will account for the extra byte or bytes, depending on implementation. If a double zero termination is implemented, then zero terminated string arrays can also be handled by the locale string implmentation.

### *Peter Knaggs*

#### Locale Wordset

Although I can see why the "locale" wordset would be of use to an application developer, I do not believe it should be incorporated in the standard. The standard should, however, provide a sufficient basis for such a wordset to be defined in a standard, or system independent, manner. The standard should not make all applications automatically multi-lingual but should allow developers the freedom to develop multi-lingual applications. I see the "locale" wordset as being at too high a level of abstraction for inclusion in the standard.

#### Character navigation

I was thinking of a single word CHAR++ ( addr -- addr ) which simply moved the pointer on to the next character. One could implement C@++ and C!++ in terms of CHAR++ with ease. I would agree with CONCAT as well, this would allow people to build the output string in a transient buffer before sending it to the output device.

I very much agree with the need for TOUPPER and TOLOWER, although I would name them >UPPER and >LOWER. The traditional method of converting a letter from upper case to lowercase, the addition of 32 is only valid in standard 7-Bit ASCII systems. For example the uppercase for the Greek letter η (eta) is H, this is not obvious from the character sets.

If you are going to advocate >UPPER and >LOWER, you should also include the >BASE, (a letter without diacritical mark). Given these conversion words we may as well provide the set of conditional functions provided by the character handling packages of other languages (such as ADA [ISO94] and C [ISO90]):

| | | | |
|---|---|---|---|
| ALPHA? | UPPER? | BASIC? | PUNCT? |
| DIGIT? | LOWER? | CNTRL? | SPACE? |

#### Bytes and Characters

I don't see any need for BMOVE or BMOVE>, given that we already have CMOVE and CMOVE> which works on a byte by byte basis. The only problem is with the name, although I believe CMOVE is acceptable given our interpretation of COUNT.

The standard does not actually allow for single byte access. Anyone wanting to use single byte access will probably be accessing memory mapped I/O, any such code can not be standard. Thus although I would agree that B! and B@ should be in the standard, there should also be a note stating that standard applications can not use them. This is similar to EKEY.

### *Nick Nelson*

I can live with this.

### ***Stephen Pelc***

### Terminology

The term locale structure is used to refer to a locale sensitive string. The present terminology is based on the implementation techniques. Perhaps we should refer to a “locale string identifier” instead, indicated by **lsid**.

### Counted strings

Counted strings are deeply ingrained in Forth. However, the way **COUNT** is currently defined makes it impossible to make their layout system dependent. This is because some users (even now) insist on abusing **COUNT** to step through memory on the assumption that a character is a byte. If the structure of a counted string is implementation dependent, then compatibility with many character sets becomes easier to implement.

For example, the Windows definition(s) of a counted string includes a 32 bit count (four bytes) of bytes, followed by that many bytes in Unicode, i.e. the character count is half the byte count.

Perhaps we should define a new type, the international string, which has these attributes.

# Change History

17 Aug 1998 – PK
A number of minor formatting changes.
Added bibliography.
Revised discussion of country codes (primary/secondary codes).
Added "Locale" wordset discussion to personal contribution.

12 Aug 1998 - SFP
Incorporated and modified many of Peter Knaggs comments.
Moved discussion comments to a separate contributers section.
Changed LINGOxxx to LOCALExxx because it sounds more official and less offensive.
Added justification for L".