ANS Forth and large characters

\\stephen\d\mpe\projects\international\i18n.widechar.v7.doc
Revised 25 March 2001

Authors:
Stephen Pelc, MicroProcessor Engineering, sfp@mpeltd.demon.co.uk
Peter Knaggs, Bournemouth University, pjk@bcs.org.uk

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

Tel: +44 (0)23 8063 1441 N.B. Area code has changed
Fax: +44 (0)23 8033 9691
Net: sfp@mpeltd.demon.co.uk
Web: http://www.mpeltd.demon.co.uk

Introduction

This document has been written to address issues raised by, but outside the scope of previous internationalisation documents. Our previous attempts to address these issues raised considerable controversy on the ANS mailing lists and on the comp.lang.forth newsgroup. We have taken the views expressed into account, and this document should be treated as a discussion document prior to an ANS proposal.

Problems

Character size

The current ANS definition of a counted string assumes that all characters are the same size, and it is implicit that this size does not change during the execution of a program. This is not true for an program designed to run with multiple languages, which may have to deal with text in 8 bit character sets (e.g. ASCII), 16 bit character sets (Unicode), and multibyte character sets (different characters may be different sizes). Multibyte character sets are used by some Forth applications in current use.

The majority of Forth applications, whether ANS or not, make the assumption that characters, bytes and address units are the same. Regardless of whether this is desirable or not, it is a fact of life that makes application portability prone to error.

Signed and unsigned characters

The current ANS specification defines **C@** as zero extending the character to fill the cell on the stack. For all character sizes less than a cell, the effect is that any future character comparison is unsigned. The wording of **COMPARE** is ambiguous (“numerically less than”) with respect to signed or unsigned comparison, and implementations of **COMPARE** that the authors have inspected do differ. This has little impact while the standard only defines a 7 bit

character set in the standard, but use even in the limited environment of Europe requires the use of 8 bit characters.

We strongly recommend that characters be defined as unsigned so that **C@** is not broken.

Counted strings

The ANS definition of a counted string declares that the count at the start of the string is the same size as a character (3.1.3.4 and 6.1.0980). This is incompatible with the use of multibyte (variable size) character sets. Similarly **CHARS** (6.1.0898) also assumes that characters are a fixed size, and **CHAR+** (6.1.0897) makes no statement about whether c-addr1 needs to contain a character.

Terminology

(This section is taken from the LOCALE proposal)

The language and character set encoding used by the Forth system at development time is referred to as the Development Character Set (**DCS**). The development character set is assumed never to change. It is furthermore assumed that character manipulation in the Forth system is defined in terms of the DCS, and that the action of character operations such as **CMOVE** is locked to the DCS.

The language and character set encoding used by any underlying operating system is referred to as the Operating Character Set (**OCS**). Note that in many environments, including embedded systems, the OCS may be the same as the DCS.

The language and character set encoding used at application run time is referred to as the Application Character Set (**ACS**). It is assumed that the largest character in an ACS fits in the native cell of the development Forth system. Note that in many environments, including embedded systems, the ACS may be the same as the DCS.

The DCS is usually seven or eight bit ASCII in the majority of today's Forth systems, but we may see Unicode systems in the near future. The OCS is defined by the host machine, and is defined by the user of the application. Thus, an application written in a Forth designed for ISO-Latin1 may be running on an O/S with a Chinese OCS, and a visitor may switch the application into yet another ACS, such as Russian. Such scenarios are rare within the US and Europe, but are common elsewhere in the world. Countries such as South Africa exist with 17 official languages, and some languages such as Portugese and English are spoken in many different countries.

Discussions

Character size

For 8 bit fixed size characters and a byte addressed machine there are few problems (except for cell addressed machines), and these are the conditions under which most existing Forth systems execute.

For 16 or 32 bit fixed size characters and a byte addressed machine, the ANS specification is consistent, but note that many of the file words are specified in terms of **characters**, including **READ-FILE** and **WRITE-FILE**. It is consequently impossible in a standard Unicode system to write an odd number of bytes to a file. There is also no operator to convert from address units to characters. Similarly the **BLOCK** word set is specified in terms of characters, which

makes text portable in terms of the 16 by 64 screen layout, but makes data non-portable unless all **BLOCK** code is made sensitive to block size.

Note that all the ANS data type conversion operators convert to Address Units (AUs), and that conversion from AUs to data size is problematic on cell addressed machines with character packing as phrases such as "**1 CHARS /"** will return incorrect information.

For multibyte (variable-sized) characters, counted strings do not work, and both **CHARS** and **CHAR+** don't work properly because they have no characters to work on. Because of the rarity of multibyte character sets, we believe that they need be handled only by the LOCALE wordset proposal for internationalisation of applications.

Machines that are not byte-addressed are mostly of larger address units, and packing characters (e.g. four 9 bit characters packed into a 36 bit cell) requires the character address to be larger (in bits) than the cell address. Such machines will then require a scheme such as that proposed in Greg Bailey's OCTET proposal.

Given the rarity of byte addressed Forths with character sizes of two or more bytes, it would seem that the least code breakage in a transition to Unicode or other larger character size will be caused with the following scenario:

1) Because of the existence of string parsing words widely used in dictionary and wordlist construction such as **PARSE**, define the ANS character as applying to the development character set (DCS). The effect of this is to leave unbroken all existing systems that the authors know of. For the reasons stated above, characters should be stated to be unsigned. From the point of view of the ANS standard document, the definition and use of the term **character** remains unchanged, although (see below) some attention to the file and block I/O words is needed.
2) Define a wide character set such that the size of a wide character is implementation dependent such that char<=widechar<=cell, and is unsigned. The WIDECHAR wordset may be constructed from existing character-oriented words by prefixing then with a 'W' character and replacing **character** by **wide character** in their definitions.
3) For Unicode systems, in which the DCS is greater than 8 bits, it well be sensible to define either an OCTET based character set or a "narrow" character set which may be defined in a similar to the widechar character set, but with an 'N' prefix.
4) Without adding several new I/O words, it appears that defining file i/o in terms of address units will break almost no existing code, except on some cell addressed machines. This can be handled by defining the "fam" words such as BIN more carefully, and perhaps adding a couple more. OCT has already been suggested for octets.

This approach is clean. Existing programs with an 8 bit DCS are unbroken, including those that use **COUNT** to step through memory. A development environment may migrate to Unicode either by making the DCS character size 16 bits, and thus breaking some code, or by retaining an 8 bit DCS, and defining the widechar as 16 bits or more.

In essence, this permits implementors to choose a migration path according to the requirements of their major applications.

Multiple character set input and output.

There will always be some strings (such as modem and RFC822 commands) that will always require 7 bit ASCII. In a Unicode native set, this will require the presence of the narrow character set, or use of (at least) the simplified OCTET proposal suggested by Greg Bailey consisting of:

```
B@ B! BMOVE B, BYTES BYTE+
```

Change history

25 March 2001

- Minor text changes

15 June 1999

- Clarified terminology section
- Removed controversy from character size discussion