

UNICODE Strings Functions

Hi All.

Researching in the source files of Harbour I found strings functions that work with UNICODE/ANSI and only ANSI.

This list may be incomplete.

Best regards,

Claudio Soto.

Functions that support UNICODE and ANSI strings

* harbour/src/rtl/chruni.c

/* Unicode(character) and Binary(byte) string functions: */

HB_UCHAR(<nCode>) -> <cText> // return string with U+nCode character in HVM CP encoding

HB_BCHAR(<nCode>) -> <cText> // return 1 byte string with <nCode> value

HB_UCODE(<cText>) -> <nCode> // return unicode value of 1-st character (not byte) in given string

HB_BCODE(<cText>) -> <nCode> // return value of 1-st byte in given string

HB_ULEN(<cText>) -> <nChars> // return string length in characters

HB_BLEN(<cText>) -> <nBytes> // return string length in bytes

HB_UPEEK(<cText>, <n>) -> <nCode> // return unicode value of <n>-th character in given string

HB_BPEEK(<cText>, <n>) -> <nCode> // return value of <n>-th byte in given string

HB_UPOKE([@]<cText>, <n>, <nVal>) -> <cText> // change <n>-th character in given string to unicode <nVal> one and return modified text

HB_BPOKE([@]<cText>, <n>, <nVal>) -> <cText> // change <n>-th byte in given string to <nVal> and return modified text

HB_USUBSTR(<cString>, <nStart>, <nCount>) -> <cSubstring>

HB_BSUBSTR(<cString>, <nStart>, <nCount>) -> <cSubstring>

HB_ULEFT(<cString>, <nCount>) -> <cSubstring>

HB_BLEFT(<cString>, <nCount>) -> <cSubstring>

HB_URIGHT(<cString>, <nCount>) -> <cSubstring>

HB_BRIGHT(<cString>, <nCount>) -> <cSubstring>

HB_UAT(<cSubString>, <cString>, [<nFrom>], [<nTo>]) -> <nAt>

HB_BAT(<cSubString>, <cString>, [<nFrom>], [<nTo>]) -> <nAt>

* harbour/src/rtl/hbtoken.c

HB_TOKENCOUNT()

HB_TOKENGET()

HB_TOKENPTR() /* like HB_TOKENGET() but returns next token starting from passed position (0 based) inside string, f.e.:

HB_TOKENPTR(cString, @nTokPos, Chr(9)) -> cToken */

HB_ATOMENS()

* harbour/src/rtl/memofile.c

MEMOREAD()

MEMOWRIT()

HB_MEMOREAD() // not limited to 64 KB as MEMOREAD()

HB_MEMOWRIT() // not limited to 64 KB as MEMOWRIT()

* harbour/src/rtl/mlcfunc.c

/* warning <nLineLength> is in bytes, <nLineLength> must be greater than the number of bytes of the longest line of text in UTF-8 */

MEMOLINE(<cString>, [<nLineLength>=79], [<nLineNumber>=1], [<nTabSize>=4], [<IWrap>=.T.], [<cEOL>|<acEOLs>]) -> <cLine>

```

MLCOUNT ( <cString>, [ <nLineLength>=79 ], [ <nTabSize>=4 ], [ <IWrap>=.T. ], [ <cEOL>|<acEOLs> ] ) ->
<nLines>
MLPOS ( <cString>, [ <nLineLength>=79 ], [ <nLineNumber>=1 ], [ <nTabSize>=4 ], [ <IWrap>=.T. ], [
<cEOL>|<acEOLs> ] ) -> <nLinePos>
/*
MLCTOPOS() // not support UTF-8
MPOSTOLC() // not support UTF-8
*/

* harbour/src/rtl/mtran.c
MEMOTRAN()

* harbour/src/rtl/replic.c
REPLICATE() /* returns n copies of given string */

* harbour/src/rtl/strc.c
HB_STRDECODEESCAPE ( <cEscSeqStr> ) -> <cStr> /* decode string with \ escape sequences */

HB_STRCDECODE ( <cStr> [, @<ICont> ] ) -> <cResult> | NIL /* decode string using C compiler rules */
/* If second parameter <ICont> is passed by reference then it allows to decode multiline strings.
In such case <ICont> is set to .T. if string ends with unclosed "" quoting.
Function returns decoded string or NIL on syntax error. */

* harbour/src/rtl/strmatch.c
HB_WILDMATCH (cPattern, cValue [, IExact] ) /* compares two strings */
/* Compares cValue with cPattern.
cPattern * may contain wildcard characters (?)
When IExact is TRUE then it will check if whole cValue is covered by cPattern
else it will check if cPattern is a prefix of cValue */

HB_WILDMATCHI (cPattern, cValue) /* compares two strings */
/* Compares cValue with cPattern
Check if whole cValue is covered by cPattern */

HB_FILEMATCH (cFileName, cPattern)
/* eg. HB_FILEMATCH ("picture.bmp", "*.bmp") ---> return TRUE if file exist */
/* eg. HB_FILEMATCH ("c:\image\picture.bmp", "picture.bmp") ---> return TRUE if file exist */

* harbour/src/rtl/strtoexp.c
HB_STRTOEXP() /* convert string to valid macrocompiler expression */

* harbour/src/rtl/strtran.c
STRTRAN()

* harbour/src/rtl/trim.c
LTRIM()
RTRIM()
TRIM() /* synonymn for RTRIM */
ALLTRIM()

* harbour/src/vm/hvm.c
/* operator $ */
<cSubStr> $ <cStr> /* return TRUE if <cSubStr> is contained in <cStr> */

* harbour/src/rtl/cdpapihb.c
HB_STRTOUTF8 ( <cStr> [, <cCPID> ] ) -> <cUTF8Str>
HB_UTF8TOSTR ( <cUTF8Str> [, <cCPID> ] ) -> <cStr>
* <cCPID> is Harbour codepage id, f.e.: "EN", "ES", "ESWIN", "PLISO", "PLMAZ", "PL852", "PLWIN", ...
* When not given then default HVM codepage (set by HB_SETCODEPAGE()) is used.

HB_TRANSLATE ( <cSrcText>, [ <cPageFrom> ], [ <cPageTo> ] ) --> cDstText /* is used usually to convert between the
Dos and the Windows code pages of the same language */

HB_UTF8CHR ( )
HB_UTF8ASC ( )

```

```
HB_UTF8AT ()
HB_UTF8RAT () /* NOTE: In HB_UTF8RAT we are still traversing from left to right, as it would be required anyway to
determine the real string length */
HB_UTF8SUBSTR ()
HB_UTF8LEFT ()
HB_UTF8RIGHT ()
HB_UTF8PEEK ()
HB_UTF8POKE ()
HB_UTF8STUFF ()
HB_UTF8LEN ()
HB_UTF8STRTRAN() /* equal to STRTRAN() */
```

/* Miscellaneous Functions */

*** -----**

```
All functions STRINGS related to DATE and TIME
SPACE() /* returns n copies of a single space */
STR()
STRZERO ()
TYPE()
VAL()
HB_VALTOSTR() /* converts any data type to STR*/
VALTYPE()
HB_ISSTRING()
HB_ISCHAR()
HB_ISMEMO()
```

Functions that support only ANSI strings

/* All these functions work with CodePages using custom character indexes (ANSI CodePage), do not support UNICODE. */

** harbour/src/rtl/at.c*
[HB_AT\(\)](#)
[AT\(\)](#)
** harbour/src/rtl/ati.c*
[HB_ATI\(\)](#)
** harbour/src/rtl/hbstrsh.c*
[HB_STRSHRINK\(\)](#)
** harbour/src/rtl/left.c*
[LEFT\(\)](#)
** harbour/src/rtl/len.c*
[LEN\(\)](#)
** harbour/src/rtl/pad.c*
[PAD\(\)](#) */* synonymn for PADR */*
** harbour/src/rtl/padc.c*
[PADC\(\)](#)
** harbour/src/rtl/padl.c*
[PADL\(\)](#)
** harbour/src/rtl/padr.c*
[PADR\(\)](#)
** harbour/src/rtl/rat.c*
[RAT\(\)](#)
[HB_RAT\(\)](#)
** harbour/src/rtl/right.c*
[RIGHT\(\)](#)
** harbour/src/rtl/strcase.c*
[LOWER\(\)](#)
[UPPER\(\)](#)
** harbour/src/rtl/stuff.c*
[STUFF\(\)](#)
** harbour/src/rtl/sub str.c*
[SUB_STR\(\)](#)
** harbour/src/rtl/transfrm.c*
[TRANSFORM\(\)](#)