Ansi Library

Table of contents

Introduction

Ansi library is an include file for Hollywood that will help you to manage ANSI escape codes so you can print colored text in the console of your host system.

I've developed this library to have an invaluable help while I'm debugging application because this way I'm able to spot on the fly error and warning messages that are hilighted from the rest of the messages.

Of course you need that your host system console is able to understand ANSI ascape codes, but almost any OS is able to do that except Windows, for which you need to install thirdy party application to accomplish the task.

I'm using ansicon program to make use of ANSI codes in my Windows development machine.

Status : Stable
Version : 1.0
Dependancies : None
License : Shareware
Author : Fabio Falcucci

Overview

Ansi Library is mapped in two root tables `Ansi` and `Term`.
`Ansi` table is used for all ANSI escape codes and strictly ANSI escape codes related functions, `Term` table is used for a very basic terminal interface providing inout and output functions for the host console.

`Ansi` Root Table

Ansi.Version	Holds the library version string
Ansi.Build	Holds the library build date
Ansi.Escape	Default ANSI escape string
Ansi.Code	A table with the most common ANSI escape sequence strings
Ansi.GetDeleteLines()	Returns the escape sequence to delete <n> lines
Ansi.GetInsertBlankLines()	Returns the escape sequence to insert <n> lines
Ansi.GetClearCharacters()	Returns the escape sequence to clear <n> characters
Ansi.GetInsertCharacters()	Returns the escape sequence to insert <n> characters
Ansi.GetDeleteCharacters()	Returns the escape sequence to delete <n> characters
Ansi.GetCursorMove()	Returns the escape sequence to move the cursor to a given location
Ansi.GetCursorUp()	Returns the escape sequence to move the cursor <n> lines up
Ansi.GetCursorDown()	Returns the escape sequence to move the cursor <n> lines down
Ansi.GetCursorRight()	Returns the escape sequence to move the cursor <n> columns right

Ansi.GetCursorLeft()	Returns the escape sequence to move the cursor <n> columns left
Term.AnsiMode	`True` or `False`, used to switch ANSI on/off
Term.NormalColor	A table that defines the terminal **normal** messages colors
Term.NoticeColor	A table that defines the terminal **notice** messages colors
Term.WarningColor	A table that defines the terminal **warning** messages colors
Term.ErrorColor	A table that defines the terminal **error** messages colors
Term.AdviceColor	A table that defines the terminal **advice** messages colors
Term.QuoteColor	A table that defines the terminal **quote** messages colors
Term.PromptColor	A table that defines the terminal **input** messages colors
Term.Size	A table used to define the terminal grid size
Term.Input()	Read input from the keyboard
Term.Print()	Print messages to the terminal
Term.Line()	Draw character line
Term.CTLine()	Draw a line with center text
Term.TEST()	Test/Demo function

Details

This section explain in details all Ansi library's variables and functions.

Ansi.Version

```
Ansi.Version
```

Holds the current library version string with the format major.minor, for example "`1.0`", "`1.1`" and so on.

Ansi.Build

```
Ansi.Build
```

Holds the current library build date string with the format dd/mm/yyyy, for example "`10/05/2014`", "`21/07/2015`" and so on.

Ansi.Escape

```
Ansi.Escape
```

Holds the default escape code string "`\27`".

Ansi.Code

`Ansi.Code`

Holds a table with the most common ANSI escape sequence strings, each sequence can be accessible also using string macros to simplify the text coloring process, note that table keys prefix by `Fg` are foreground colors, while table keys prefix by `Bg` are background colors.

Table Key	String Macro
Reset	~{RSET}
BoldOn	~{BLD+}
DimOn	~{DIM+}
ItalicsOn	~{ITA+}
UnderlineOn	~{UND+
BlinkOn	~{BLI+}
InverseOn	~{INV+}
HiddenOn	~{HID+}
StrikethroughOn	~{STR+}
BoldOff	~{BLD-}
ItalicsOff	~{ITA-}
UnderlineOff	~{UND-}
BlinkOff	~{BLI-}
InverseOff	~{INV-}
HiddenOff	~{HID-}
StrikethroughOff	~{STR-}
FgBlack	~{FBLK}
FgRed	~{FRED}
FgGreen	~{FGRN}
FgYellow	~{FYEL}
FgBlue	~{FBLU}
FgMagenta	~{FMAG}
FgCyan	~{FCYA}
FgWhite	~{FWHI}
FgDefault	~{FDEF}
BgBlack	~{BBLK}
BgRed	~{BRED}
BgGreen	~{BGRN}
BgYellow	~{BYEL}
BgBlue	~{BBLU}
BgMagenta	~{BMAG}
BgCyan	~{BCYA}
BgWhite	~{BWHI}
BgDefault	~{BDEF}
MoveCursor	
moveCursorUp	
MoveCursorUp1	~{M1UP}
MoveCursorDown	
MoveCursorDown1	~{M1DO}

MoveCursorRight	
MoveSursorRight1	~{M1RI}
MoveCursorLeft	
MoveCursorLeft1	~{M1LE}
SaveCursor	
RestoreCursor	~{REST}
Home	~{HOME}
ClearBelow	~{CLRB}
ClearAbove	~{CLRA}
ClearHome	~{CLRH}
ClearLineRight	~{CLRR}
ClearLineLeft	~{CLRL}
ClearLine	~{CLIN}
ClearCharacters	
Clear1Character	~{C1CH}
InsertBlankLines	
Insert1BlankLine	~{I1BL}
DeleteLines	
Delete1Line	~{D1LI}
DeleteCharacters	
Delete1Character	~{D1CH}
InsertCharacters	
Insert1Character	~{I1CH}
ShowCursor	~{SHWC}
HideCursor	~{HIDC}

Ansi.GetDeleteLines()

```
eSeq = Ansi.GetDeleteLines(n)
```

This function generates and returns the ANSI escape code sequence string needed to delete `n` lines from the terminal.

Ansi.GetInsertBlankLines()

```
eSeq = Ansi.GetInsertBlankLines(n)
```

This function generates and returns the ANSI escape code sequence string needed to insert `n` blank lines to the terminal.

Ansi.GetClearCharacters()

```
eSeq = Ansi.GetClearCharacters(n)
```

This function generates and returns the ANSI escape code sequence string needed to clear `n` characters from the terminal.

Ansi.GetInsertCharacters()

```
eSeq = Ansi.GetInsertLines(n)
```

This function generates and returns the ANSI escape code sequence string needed to insert `n` characters to the terminal.

Ansi.GetDeleteCharacters()

```
eSeq = Ansi.GetDeleteCharacters(n)
```

This function generates and returns the ANSI escape code sequence string needed to delete `n` characters from the terminal.

Ansi.GetCursorMove()

```
eSeq = Ansi.GetCursorMove(row, column)
```

This function generates and returns the ANSI escape code sequence string needed to move the cursor at the position (row, column) in the terminal.

Ansi.GetCursorUp()

```
eSeq = Ansi.GetCursorUp(n)
```

This function generates and returns the ANSI escape code sequence string needed to move the cursor up `n` lines in the terminal.

Ansi.GetCursorDown()

```
eSeq = Ansi.GetCursorDown(n)
```

This function generates and returns the ANSI escape code sequence string needed to move the cursor down `n` lines in the terminal.

Ansi.GetCursorRight()

```
eSeq = Ansi.GetCursorRight(n)
```

This function generates and returns the ANSI escape code sequence string needed to move the cursor right `n` columns in the terminal.

Ansi.GetCursorLeft()

```
eSeq = Ansi.GetCursorLeft(n)
```

This function generates and returns the ANSI escape code sequence string needed to move the cursor left `n` columns in the terminal.

Term.AnsiMode

```
Term.AnsiMode
```

This variable is used to parse or ignore ANSI escape sequences by the terminal functions used to print messages to the screen.

`True` ANSI is parsed and interpreted
`False` ANSI is ignored and discarted

Messages Colors

Ansi library provide some string macros you can use to simplify the process of text coloring, each message type is defined in the root table `Term` as follow:

String Macro	Description	Default Colors	Table Key
~{NORM}	Normal messages	White on Black	NormalColor
~{NOTI}	Notice messages	Cyan on Black	NoticeColor
~{WARN}	Warning messages	Yellow on Black, Bold	WarningColor
~{ERRO}	Error messages	Yellow on Red, Bold	ErrorColor
~{ADVI}	Advice messages	Green on Black, Bold	AdviceColor
~{QUOT}	Quoted text	Red on Black, Bold + Italic	QuoteColor
~{PROM}	Prompt (input messages)	Magenta on Black, Bold + Italic	PromptColor

Each table item have the same structure as follow:

```
{ Fg = AnsiColor,
  Bg = AnsiColor,
  Bold = Bool,
  Italic = Bool,
  Underline = Bool }
```

For example, the normal messages color (NormalColor), is defined this way:

```
{ Fg = Ansi.Code.BgBlack,
  Bg = Ansi.Code.FgWhite,
  Bold = False,
  Italic = False,
  Underline = False }
```


Term.Size

```
Term.Size
```

This table defines the default terminal size, it holds the two items `rows` and `columns` that with their respective default values of `25` and `80`.

Term.Input()

```
result = Term.Input(t, wordwrap, linefeed)
```

This function is used to read input from the terminal, the user must terminate the input using the `ENTER` key.

ARGUMENTS

`t`	Message for the user. ANSI macros are allowed.
`wordwrap`	`True` to enable the wordwrap
`linefeed`	`True` to add a line feed at the end of the message

Term.Print()

```
Term.Print(t, wordwrap, newlineoffset, linefeed)
```

This function is used to print messages to the terminal.

ARGUMENTS

`t`	Message to print. ANSI macros are allowed.
`wordwrap`	`True` to enable the wordwrap (default `True`)
`newlineoffset`	Right offset used to align text when it wraps (when wordwrap is enabled)
`linefeed`	`True` to add a line feed at the end of the message (default `True`)

Term.Line()

```
Term.Line(character, style)
```

This function is used to print a line with the given `character` and the specified `style`.

ARGUMENTS

`character`	The character to use to generate the horizontal line
`style`	Ansi style to use, you can also use the string macros to set the style

Term.CTLine()

```
Term.CTLine(character, text, style)
```

This function is used to print a line with the given `character` and the specified `style`. The `text` will be centered.

ARGUMENTS

`character`	The character to use to generate the horizontal line
`text`	The text to center
`style`	Ansi style to use, you can also use the string macros to set the style

Term.TEST()

```
Term.TEST()
```

Test function you can use as an example and for testing your terminal capabilities.