7 Console Display Control (console.hhf)

The HLA console module provides a reasonably portable way to control the console display under different operating systems. The routines in this module let you write "really-smart-terminal" console applications that behave in a similar fashion under different operating. The HLA console module routines take advantage of the Windows console API when running under Windows, they use the VT100/ANSI terminal control code sequences for other operating systems that use ANSI terminal control codes for console control. The routines in this module let you control the cursor position, erase selected portions of data from the screen, insert and delete characters and lines of text, scroll the screen, select display colors, and so on.

Note: this console module replaces the older HLA Standard Library console module that was Win32-specific. That earlier console module provided many features that are not present in the current console module because the Win32 console capabilities are quite a bit more sophisiticated than is possible with an ANSI terminal emulation. Older code that took advantage of these extra features will not be able to compile properly with this new console module. The original console module is still available in the HLA distribution under the "Examples" directory; new code, however, should not use that module; if you want to take advantage of the Win32 console capabilities, you should call the Win32 API routines directly.

Note: be sure to read the chapter on "Passing Parameters to Standard Library Routines" (parmpassing.rtf) before reading this chapter.

A Note About Thread Safety: The args module maintains a couple of static global variables that maintain the command-line values. Currently, these values apply to all threads in a process. You should take care when changing these values in threads. The command-line is a resource that must be shared amongst all threads in an application. If you write multi-threaded applications, it is your responsibility to serialize access to the command-line functions.

7.1 The Console Module Module

To use the date functions in your application, you will need to include one of the following statements at the beginning of your HLA application:

```
#include( "console.hhf" )
or
#include( "stdlib.hhf" )
```

7.2 Cursor Positioning Functions

The functions in this category reposition the cursor on the display

```
procedure console.gotoxy( x:dword; y:dword );@pascal
procedure console.gotorc( r:dword; c:dword );@stdcall
```

console.gotoxy and *console.gotorc* are actually the same function. The only difference between the two calls is that they (internally) swap their parameters before making the call to the function. Note that the "rc" in gotorc stands for "row/column" which is equivalent to saying "gotoyx". The console.gotorc function was provided because people intuitively prefer to specify the row (y) value as the first argument and the column (x) value as the second argument.

These functions position the cursor at the specified (x,y)/(c,r) coordinate on the screen.

```
HLA high-level calling sequence examples:

  console.gotoxy( 0, 10 );
  stdout.put( "Print this on line 10, column 0" nl );
  console.gotoxy( 15, 0 );
  stdout.put( "Print this on line 15, column 0" nl );


HLA low-level calling sequence examples:
```

```
    pushd( 0 );
    pushd( 10 );
    call console.gotoxy;// row = 10, column = 0

    // Note that console.gotorc uses the @stdcall calling convention, so
    // it's arguments are reversed from the declaration, that is, you
    // push the same exact arguments you push for gotoxy (which makes
    // sense, as both functions are actually the same code).

    pushd( 0 );
    pushd( 15 );
    call console.gotorc;
```

procedure console.up();

console.up moves the cursor up one line. Because of the variation in terminal emulations out there, the results are undefined if you execute this procedure when the cursor is on the first line of the display. Some consoles scroll the screen down one line, others ignore the request.

HLA high-level calling sequence example:

```
    console.up();
```

HLA low-level calling sequence example:

```
    call console.up;
```

procedure console.nup(n:uns32);

console.nup moves the cursor up n lines. Because of the variation in terminal emulations out there, the results are undefined if this procedure attempts to move above the top line on the display. Some consoles scroll the screen down one line, others ignore the request.

HLA high-level calling sequence example:

```
    console.nup( 5 );
```

HLA low-level calling sequence example:

```
    pushd( 5 );
    call console.nup;
```

procedure console.down();

console.down moves the cursor down one line. Because of the variation in terminal emulations out there, the results are undefined if you execute this procedure when the cursor is on the last line of the display. Some consoles scroll the screen up one line, others ignore the request..

HLA high-level calling sequence example:

```
    console.down();
```

HLA low-level calling sequence example:

```
    call console.down;
```

procedure console.ndown(n:uns32);

console.ndown moves the cursor down one line. Because of the variation in terminal emulations out there, the results are undefined if you execute this procedure when the cursor is on the last line of the display. Some consoles scroll the screen up one line, others ignore the request.

```
HLA high-level calling sequence example:

  console.ndown( 5 );


HLA low-level calling sequence example:

  pushd( 5 );
  call console.ndown;
```

procedure console.left();

console.left moves the cursor up n lines. Because of the variation in terminal emulations out there, the results are undefined if you execute this procedure and it attempts to move the cursor before the first column on the line. Some consoles move the cursor to the end of the previous line, others ignore the request.

```
HLA high-level calling sequence example:

  console.left();


HLA low-level calling sequence example:

  call console.left;
```

procedure console.nleft(n:uns32);

console.nleft moves the cursor down one line. Because of the variation in terminal emulations out there, the results are undefined if you execute this procedure when the cursor is on the last line of the display. Some consoles scroll the screen up one line, others ignore the request.

```
HLA high-level calling sequence example:

  console.nleft( 5 );


HLA low-level calling sequence example:

  pushd( 5 );
  call console.nleft;
```

procedure console.right();

console.right moves the cursor to the right one character position. Because of the variation in terminal emulations out there, the results are undefined if you execute this procedure when the cursor is at the last column of the display. Some consoles move the cursor to the beginning of the next line, others ignore the request.

```
HLA high-level calling sequence example:

  console.right();


HLA low-level calling sequence example:

  call console.right;
```

procedure console.nright(n:uns32);

console.nright moves the cursor down one line. Because of the variation in terminal emulations out there, the results are undefined if you execute this procedure and it attempts to move the cursor beyond the last column of the display. Some consoles move the cursor to the beginning of the next line, others ignore the request.

```
HLA high-level calling sequence example:

  console.nright( 5 );


HLA low-level calling sequence example:

  pushd( 5 );
  call console.nright;
```

procedure console.saveCursor();

console.saveCursor saves the current cursor position in an internal variable. You can restore the cursor position via the *console.restoreCursor* call. Note that there is only one level of save available.

```
HLA high-level calling sequence example:

  console.saveCursor();


HLA low-level calling sequence example:

  call console.saveCursor;
```

procedure console.restoreCursor();

console.saveCursor restores the cursor to the position previously saved by *console.saveCursor*. Note that there is only one level of "save" available.

```
HLA high-level calling sequence example:

  console.restoreCursor();


HLA low-level calling sequence example:

  call console.restoreCursor;
```

7.3 Console Clearing Functions

The functions in this category clear portions (or all) of the display.

procedure console.cls();
procedure console.home();

console.cls and *console.home* are actually the same function. These procedures clear the screen and move the cursor to the home (0,0) position.

```
HLA high-level calling sequence examples:

  console.cls();
  console.home();


HLA low-level calling sequence examples:

  call console.cls;
  call console.home;
```

procedure console.clrToEOLN();

console.clrToEOLN clears the text (by writing spaces) from the current cursor position to the end of the line that the cursor is on.

```
HLA high-level calling sequence examples:

  console.clrToEOLN();

HLA low-level calling sequence examples:

  call console.clrToEOLN;
```

procedure console.clrToBOLN();

console.clrToBOLN clears the text (by writing spaces) from the current cursor position to the beginning of the line that the cursor is on.

```
HLA high-level calling sequence examples:

  console.clrToBOLN();

HLA low-level calling sequence examples:

  call console.clrToBOLN;
```

procedure console.clrLn();

console.clrLn clears the line that the cursor is on by writing spaces to that line. This does not delete the line from the screen, it only clears the characters from the line.

```
HLA high-level calling sequence examples:

  console.clrLn();

HLA low-level calling sequence examples:

  call console.clrLn;
```

procedure console.clrToEOScrn();

console.clrToEOScrn clears the text (by writing spaces) from the current cursor position to the end of the screen.

```
HLA high-level calling sequence examples:

  console.clrToEOScrn();

HLA low-level calling sequence examples:

  call console.clrToEOScrn;
```

procedure console.clrToBOScrn();

console.clrToBOScrn clears the text (by writing spaces) from the current cursor position to the beginning of the screen.

```
HLA high-level calling sequence examples:

  console.clrToBOScrn();

HLA low-level calling sequence examples:

  call console.clrToBOScrn;
```

7.4 Character Insertion/Removal Functions

The functions in this category insert and delete characters on the console display.

procedure console.insertChar();

console.insertChar inserts room for a single character by shifting the characters under the cursor and to the right of the cursor to the right one position. The vacated position is filled with a space. The last character on the line is lost.

```
HLA high-level calling sequence examples:

  console.insertChar();


HLA low-level calling sequence examples:

  call console.insertChar;
```

procedure console.insertChars(n:dword);

console.insertChars inserts room for n characters by shifting the characters under the cursor and to the right of the cursor right n positions. The vacated positions are filled with spaces. The last n characters on the line are lost.

```
HLA high-level calling sequence examples:

  console.insertChars( n );


HLA low-level calling sequence examples:

  pushd( n );
  call console.insertChars;
```

procedure console.insertLine();

console.insertLine inserts a blank line before the line the cursor is on by pushing the line under the cursor, and the lines below the cursor, down one line on the screen. The new line is filled with blanks. The last line on the screen is lost.

```
HLA high-level calling sequence examples:

  console.insertLine();


HLA low-level calling sequence examples:

  call console.insertLine;
```

procedure console.insertLines(n:dword);

console.insertLines opens up n new blank lines at the current cursor position by pushing the lines at and below the cursor down n lines on the screen. The last n lines on the screen will be lost.

```
HLA high-level calling sequence examples:

  console.insertLines( 5 );


HLA low-level calling sequence examples:

  pushd( 5 );
  call console.insertLines;
```

procedure console.deleteChar();

console.deleteChar deletes the character under the cursor by shifting the characters after the cursor one position to the left. The last character position at the end of the line is filled with a blank.

```
HLA high-level calling sequence examples:

  console.deleteChar();


HLA low-level calling sequence examples:

  call console.deleteChar;
```

procedure console.deleteChars(n:dword);

console.deleteChars deletes n characters under and to the right of the cursor by shifting the characters after the cursor n positions to the left. The n character positions at the end of the line are filled with blanks.

```
HLA high-level calling sequence examples:

  console.deleteChars( n );


HLA low-level calling sequence examples:

  pushd( n );
  call console.deleteChars;
```

procedure console.deleteLine();

console.deleteLine deletes the line the cursor is one by shifting all the lines below the cursor position up one line. The last line on the screen is filled with blanks.

```
HLA high-level calling sequence examples:

  console.deleteLine();


HLA low-level calling sequence examples:

  call console.deleteLine;
```

procedure console.deleteLines(n:dword);

console.deleteLines procedure deletes n lines at and below the current cursor position. The vacated lines at the bottom of the screen are filled with blanks.

```
HLA high-level calling sequence examples:

  console.deleteLines( 5 );


HLA low-level calling sequence examples:

  pushd( 5 );
  call console.deleteLines;
```

7.5 Console Scrolling

The functions in this category scroll the screen up and down.

procedure console.scrollUp( );

console.scrollUp scrolls the entire screen up one line.

```
HLA high-level calling sequence examples:
```

```
  console.scrollUp();
```

HLA low-level calling sequence examples:

```
  call console.scrollUp;
```

procedure console.scrollDown( );

console.scrollDown scrolls the entire screen down one line.

HLA high-level calling sequence examples:

```
  console.scrollDown();
```

HLA low-level calling sequence examples:

```
  call console.scrollDown;
```

7.6 Console Output Colors

The functions in this category control the color of the characters printed on the display.

procedure console.setAttrs(foreground:uns32; background:uns32);

console.setAttrs sets the console internal attribute value to be used for all following character output. Use the routine to set the color of the characters you wish to print. The foreground parameter sets the color for the text characters, the background parameter sets the color of the background area of each character cell.

The console module defines the following constants that represent the corresponding colors:

```
console.black := 0;
console.red := 1;
console.green := 2;
console.yellow := 3;
console.blue := 4;
console.magenta := 5;
console.cyan := 6;
console.white := 7;
```

HLA high-level calling sequence examples:

```
  console.setAttrs( console.yellow, console.blue );
```

HLA low-level calling sequence examples:

```
  pushd( console.yellow );
  pushd( console.blue );
  call console.setAttrs;
```