AVR244: AVR UART as ANSI Terminal Interface

8-bit AVR® Microcontroller

Application Note

Features

- **Make use of standard terminal software as user interface to your application.**
- **Enables use of a PC keyboard as input and ascii graphic to display status and control information.**
- **Drivers for ANSI/VT100 Terminal Control included.**
- **Interactive menu interface included.**

Introduction

This application note describes some basic routines to interface the AVR to a terminal window using the UART (hardware or software). The routines use a subset of the ANSI Color Standard to position the cursor and choose text modes and colors. Routines for simple menu handling are also implemented.

The routines can be used to implement a human interface through an ordinary terminal window, using interactive menus and selections. This is particularly useful for debugging and diagnostics purposes. The routines can be used as a basic interface for implementing more complex terminal user interfaces.

To better understand the code, an introduction to ‘escape sequences’ is given below.

Escape Sequences

The special terminal functions mentioned (e.g. text modes and colors) are selected using ANSI escape sequences. The AVR sends these sequences to the connected terminal, which in turn executes the associated commands. The escape sequences are strings of bytes starting with an escape character (ASCII code 27) followed by a left bracket ('['). The rest of the string decides the specific operation.

For instance, the command '1m' selects bold text, and the full escape sequence thus becomes 'ESC[1m'. There must be no spaces between the characters, and the commands are case sensitive. The various operations used in this application note are described below.

Rev. 2547A–AVR–11/03

Implemented Sequences

The following terminal functions are implemented:

Function Name	Description
Term_Initialise	Resets all text modes, clears the screen and positions the cursor in the top-left corner. All is done using escape sequences.
Term_Send	Sends one byte to the attached terminal.
Term_Get	Waits and gets one byte from the attached terminal.
Term_Get_Sequence	Decodes an incoming escape sequence. Currently only supports arrow keys. It actually recognizes received sequences equal to moving one cell in any of the four directions.
Term_Send_FlashStr	Sends a 0-terminated string contained in Flash memory to the terminal one character at a time.
Term_Send_RAMStr	Sends a 0-terminated string contained in SRAM to the terminal one character at a time.
Term_Erase_ScreenBottom	Clear all lines from current line to bottom of screen.
Term_Erase_ScreenTop	Clear all lines from current line to top of screen.
Term_Erase_Screen	Clear entire terminal screen.
Term_Erase_to_End_of_Line	Clear all characters from current position to end of line.
Term_Erase_to_Start_of_Line	Clear all characters from current position to start of line.
Term_Erase_Line	Clear all characters on current line.
Term_Set_Display_Attribute_Mode	Set desired text mode. Available text modes are listed below.
Term_Set_Display_Colour	Set foreground or background to desired colour. Applies to all consecutive outputs until next change. Available colors are listed below.
Term_Set_Cursor_Position	Set current cursor position.
Term_Move_Cursor	Move cursor any number of steps in the desired direction.
Term_Save_Cursor_Position	Save current position for later retrieval. More on this below.
Term_Restore_Cursor_Position	Restore previously saved cursor position. More on this below.
Term_Set_Scroll_Mode_All	Enable scrolling for entire screen when issuing a newline on the last line of the screen.
Term_Set_Scroll_Mode_Limit	Limit scrolling window to only a few lines of the screen.
Term_Print_Screen	Issue a ‘print screen’ command to the terminal window.
Term_Draw_Frame	Display a single og double menu frame using graphical characters. This is a user-defined routine using escape sequences for drawing the frame.
Term_Draw_Menu	Display a pop-up menu with frame and highlighted choice. The menu is defines as a text string with newline-separated choices. This is a user-defined routine using escape sequences for drawing the frame and displaying the menu choices.
Term_Handle_Menu	Display a meny and handle arrow keys until Enter is pressed. Returns the choice number. This is a user-defined routine using the two routines above for handling a menu.

Text Modes

The example above (’ESC[1m’) selects bold text, which is just one of many text modes available. All text mode commands ends with the *m*-character. The following text modes are defined in the code:

Mode Number	Description
0	Plain text
1	Bold text
2	Dim text
4	Underlined text
5	Blinking text
7	Reversed text
8	Concealed text

Multiple modes can be combined using a semicolon. The sequence ’ESC[1;4m’ will select bold underlined text. The predefined functions in this application note does not support multiple modes at a time. Use subsequent function calls to set multiple modes.

Note that it is not possible to disable single modes. All modes must be canceled in one operation by selecting ‘Plain text’-mode. An example follows:

Mode command issued	Displayed text will be…
0 (Plain text)	Plain
1 (Bold text)	Bold
4 (Underlined text)	Bold and underlined
0 (Plain text)	Plain
4 (Underlined text)	Underlined only

Text Colors

The following colors are available:

Mode number	Text colour	Mode number	Background colour
30	Black	40	Black
31	Red	41	Red
32	Green	42	Green
33	Yellow	43	Yellow
34	Blue	44	Blue
35	Magenta	45	Magenta
36	Cyan	46	Cyan
37	White	47	White

These modes can also be combined, even with the other text modes. For instance, the sequence ‘ESC[4;34;40m’ will select underlined blue text on black background. The predfined functions in this application note sets either the foreground or background colour at a time.

Saving and Restoring the Cursor Position

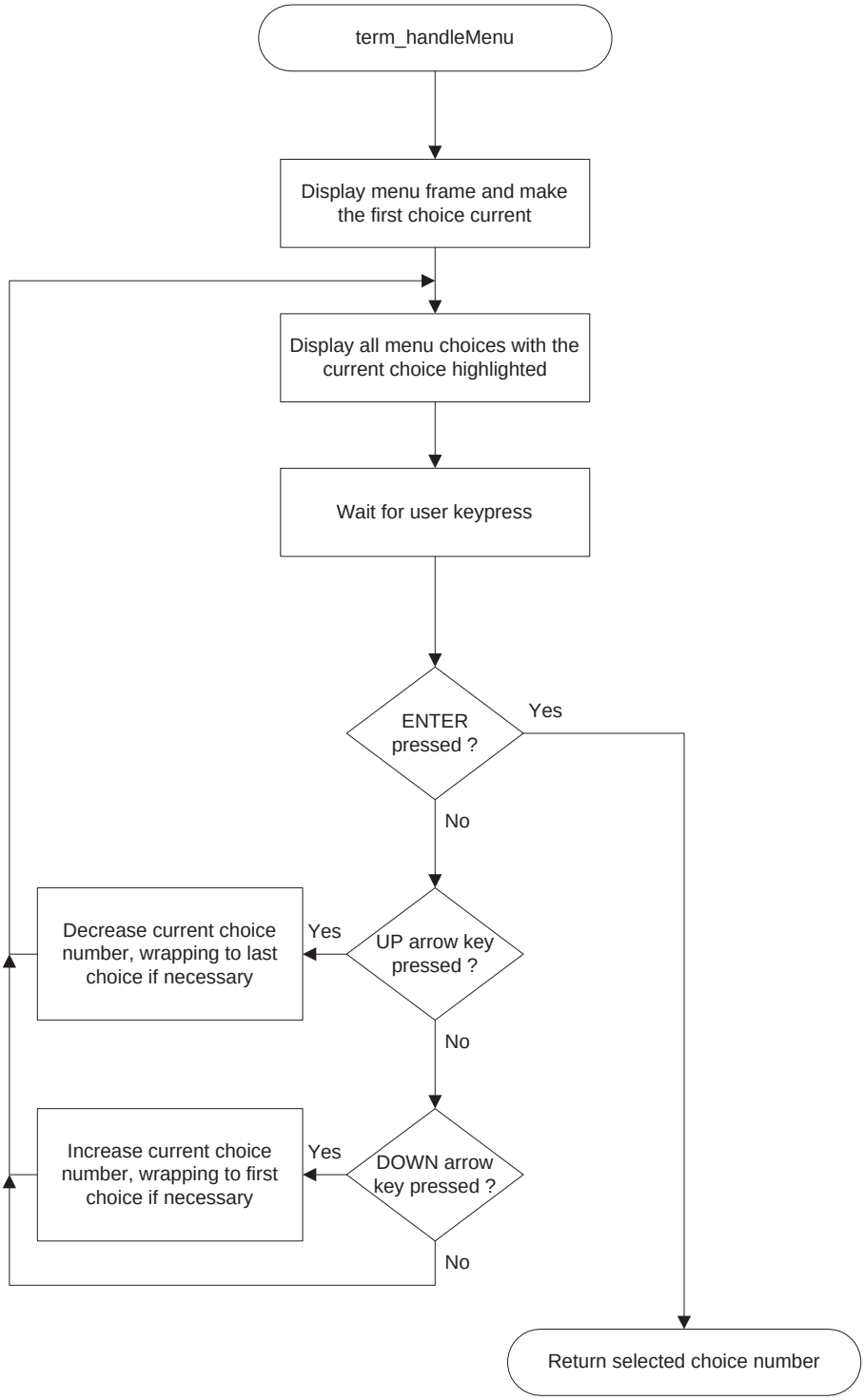
It is possible to save the current cursor position for later retrieval. This is useful when a routine that writes to an absolute window position is called from different places in the code. The cursor position can be saved prior to calling the routine, and restored afterwards.

It is not possible to save several positions. Each save operation replaces any previously saved positions. Subsequent restore operations will always return to the last saved position.

The sequence ‘ESC[s’ saves the current position and the sequence ‘ESC[u’ positions the cursor on the last saved position. The predefines functions will send the escape sequences for you.

Term_Handle_Menu Function

The flowchart for the term_handleMenu function is as follows:



Terminal Demo Application

The included demo application clears the screen and displays a three-choice menu with a double frame. It then waits for arrow key-press for moving the menu highlight or Enter for selecting a menu choice. When a choice is selected, the letter A, B or C is displayed according to the choice, and the application then echoes the user input forever.

The terminal interface is implemented as a standalone code module, easily included in other applications. There are more commands available in the ANSI standard. Only the most common are implemented in this interface, but it should be easy to implement other required operations in the code.

Terminal Software

To connect to the AVR Terminal interface, a terminal emulator program is needed. There are many programs available for doing this, one is the HyperTerminal application included in most version of Microsoft Windows®.

However, there are other programs available as freeware and shareware on the Internet. One of them is the TeraTerm® application, which can be downloaded from the following URL:

http://hp.vector.co.jp/authors/VA002416/teraterm.html

More information on the ANSI Colour Standard can be found at the following URL:

http://www.termsys.demon.co.uk/vtansi.htm