Contents

Chapter 1

Introduction

1.1 Welcome

Welcome to ProANSI, a flexible ansi editor for the Commodore *AMIGA*. I started writing this program last year, because I was not satisfied with the existing programs. They all had their strong points, but neither of them was complete. Because I desparately needed a good program for the design of all menus for my BBS I wrote ProANSI.

I originally started developing ProANSI on an A2000C under 1.3, but lateron I switched to my A500 under 2.04. Therefore I decided to design the program so that it will run both under 1.x and under 2.04, and use as many features of each OS as possible. Recently I've upgraded to the A3000/25 and continued the development on this machine.

1.2 ShareWare

This program is brought to you under the concept of ShareWare. This means that if you like this program after using it for a limited period you must register it. You can contact me through my BBS: "The Empire Strikes Back", which is situated in Delft, The Netherlands. You can reach it by calling: (0)15-147099. The BBS will also be used as support BBS for ProANSI. Alternatively you can reach me at NLA-NET 14:100/501.2 (matrix netmail).

For more information on registering please read the "Registration.doc" which is supplied with this archive. All legal stuff can also be found there.

Chapter 2

Installation

Installing ProANSI is really simple. You just put it on your harddisk and it will work. In your LIBS: directory you will need the following libraries:

- "diskfont.library"
- "req.library" [1]
- "rexxsyslib.library"
- "asl.library" [2]
- "utility.library" [3]

Optionally, you can use an IBM style font. The (hardcoded) default font is the "IBM.font", but you can easily change this in the preferences, so in fact, you can use any 8 by 8 fixed-width font that is in your FONTS: directory.

ProANSI can be started from a Shell or from the Workbench. No arguments or tool types can be added. Everything should work fine with the standard stack size, so you don't have to change that either. ProANSI works both under PAL and NTSC, and will open its screen accordingly. PAL systems have a vertical resolution of 30 lines, NTSC systems have 23 at present.

[1] Under 1.xx only!
[2] Under 2.04 only!
[3] Under 2.04 only!

Chapter 3

Working with ProANSI

It is very easy to work with ProANSI, because of the fact that the program sticks closely to the style guide.

3.1 The Keyboard

A lot of things (if not everything) can be done by using the keyboard, which was done on purpose, because creating a screen forces you to use the keyboard anyway. Therefore, almost every menu option also has a keyboard shortcut. When ProANSI reads a key from the keyboard, it always takes into account the current keymap, so this program can be used with any keymap (so everybody should be happy :-)

3.1.1 Cursor Movement

The cursor can be moved in several ways. If you type a character, the cursor will automatically move right one position. Typing *Return* forces the cursor to the beginning of the next line. Furthermore, you can move the cursor with the cursor keys. If you hold down the *SHIFT* key together with the cursor movement keys, the cursor will move to the boundary of the screen. Finally, you can move it with the mouse by pressing the left mouse button.

3.1.2 Function Keys

With the ten function keys you can select one of ten different keyblocks. The current keyblock is shown in the bottom right area of the status window.

3.1.3 The numeric keyblock

These keys can be defined by the user. As mentioned before, the current keyblock is shown in the status window. The keyblocks were designed so that you can easily access all the (otherways hard to reach) graphics characters.

3.1.4 The ESCape Key

With this key you can bring back the status window (to front) if you've clicked it to back with the appropriate gadget. This mechanism was added to allow you to view the screen completely.

3.1.5 Backspace and DELete

Both keys work like they should. Remember that neither backspace nor DELete will move the rest of the line if you are not in insert mode. The backspace key can be made destructive or not. If it is not, and insert mode is off, the cursor will only go left one character.

3.2 The Menus

The menus work as with any normal program. As many menu options as possible were given a keyboard shortcut, so you don't need to grab the mouse every time you want to select a menu item.

3.2.1 Project

This menu allows you to control the project you're working on, so this is where you can for example load or save your work.

- `New`

  Starts a new project, which totally erases the project you were working on. This action cannot be undone by using the undo option, so make sure your project was saved before selecting this option.

- `Open...`

  Loads a new project from disk. The current project will be erased. If you want to cancel this operation, you can click the 'Forget It' gadget in the file requester. Nothing will be erased when you do this.

- `Include...`

  Adds a project to the current one. It starts inserting text from the current cursor position, and if it encounters a clearscreen code, this will be converted to 'cursor home' and the screen will not be cleared.

  Keep in mind that the `No Background` and `No Foreground` flags will stay in effect when loading a picture. This allows you to instantly filter colors from a file.

- `Save`

  - `Ascii`

    The screen is saved as pure (extended) ascii. All color information is lost, and no cursor movement instructions will be used.

  - `Ansi`

    The screen is saved as an ansi screen, including colors. I've tried to keep this routine more or less ANSI X3.64 – 1979 compatible.

  - `Ansi Anim`

    The screen is saved as an ansi animation. The program remembers everything you enter, and can therefore build an ansi animation from that information.

- `Save As`

  - `Ascii...`

    Same as under `Save` only that here you will always be asked to enter a filename.

  - `Ansi...`

    Same as under `Save` only that here you will always be asked to enter a filename.

  - `Ansi Anim...`

    Same as under `Save` only that here you will always be asked to enter a filename.

- `Preview`

  Shows how the screen will be built if you save it. In case of an ansi animation, you will see the screen as you've entered it. Otherwise, the screen will be rebuilt top to bottom.

- `About...`

  Shows you the current version number of the program.

- `Quit`

  Ends the program after asking for a conformation that this is really what you want.

3.2.2 Edit Normal

Here you can perform all edit operations when in normal mode. When you are in animation mode, this menu will be ghosted which means you can't select any items.

For most items you will have to select an area with your mouse first. You do this by holding down the left mouse button and dragging the mouse across the area you want to select.

- `Mark Block`

  This item allows you to mark a block with your cursor keys. This is an alternative to the mouse drag method, for those of you who don't want to reach for the mouse every time they want to mark a block.

- `Cut Block`

  Cuts out the selected area. The contents will be put in the copy buffer. The empty space is filled with spaces (in foreground color 7, background color 0).

- `Copy Block`

  Copies the selected area to the copy buffer.

- `Paste Block`

  Pastes the area in the copy buffer back to the screen. The top left corner is determined by the position of the cursor.

- `Box Block`

  Draws a box around the selected area. The box will be drawn using the currently selected keyblock, using the 7 as the top-left, 9 as top-right, 1 as bottom-left and 3 as bottom-right character, the 0 for horizontal lines and the *Enter* key for vertical lines.

- `Cut Line`

  Cuts the line on which the cursor is standing out of the screen. The rest of the screen will be moved up one line, and the line you cut out is kept in the so called line buffer.

- `Copy Line`

  Copies the line the cursor is standing on to the line buffer.

- `Paste Line`

  Pastes the line from the line buffer to the line the cursor is currently on. The rest of the screen is moved down one line, and the last line of the screen is therefore lost.

- `Delete Line`

  Deletes the line the cursor is on. It will not be removed but is filled with spaces.

3.2.3 Edit Anim

In this menu you'll find all edit operations that can be performed when in animation mode. Most of them have something to do with the undo/redo buffer, which is used when making an animation. If you're not in animation mode, this entire menu will be ghosted, and therefore no items can be selected.

- `Undo`

  With the `Undo` function you can take back your edit operations step by step. If you undo something it will not be lost, so you can also undo some steps, then add some new, and finally redo the steps you'd undone.

- `Undo All`

  Simply repeats the `Undo` operation until it arrives at the blank (black) screen you started with.

- `Zap Undo`

  This operation works like the `Undo` option except for the fact that a `Zap Undo` can't be taken back by the `Redo` option (it is gone, therefore the name 'zap').

- `Redo`

  With the 'redo' function you can redo everything you've removed with 'undo'.

- `Redo All`

  Calls the `Redo` function until it arrives at the end of the undo/redo buffer.

- `Kill Redo Buffer`

  Kills all characters in the redo buffer. This is useful when you want to delete the 'rest' of an animation.

3.2.4 Color

This menu allows you to do almost anything with the colors of your project. You can recolor characters or blocks, and lots of other useful things.

- `No Foreground`

  If this flag is enabled, you can't overwrite the foreground color of the characters, which means that it will remain the same. This is handy when you want to change some text in a multicolor screen, without having to worry about (re)selecting colors all the time. Also, you could filter the foreground colors out of a screen by enabling this option, and subsequently loading an ansi screen.

- `No Background`

  If this flag is enabled, you can't overwrite the background color of the characters. The same rules apply as with `No Foreground`.

- `Get`

  - `Foreground`

    Grabs the foreground color under the cursor, and selects it as the current foreground color.

  - `Background`

    Grabs the background color under the cursor, and selects it as the current background color.

  - `Both`

    Grabs the fore- and background color under the cursor, and selects them as the current colors.

- `Set`

  If a block was marked, these functions apply to the whole block, if not, they apply to the character under the cursor.

  - `Foreground`

    Changes the foreground color.

  - `Background`

    Changes the background color.

– `Both`

Changes both the foreground and the background color.

3.2.5 Prefs

In this menu, you can configure the program to your own needs. All preference settings can be saved under a selectable name. At startup, the file "ProANSI.prefs" is automatically loaded, if found in the current directory[1].

- `Open...`

  Opens new preferences. Be careful when using this option, because it will erase your current project. Of course, you'll be notified of this, so you can abort the operation if you selected it be accident.

- `Save`

  Saves the preferences under the 'current' name.

- `Save As...`

  Saves the preferences under a name you can select using a file (save) requester. As a convention you should use '.prefs' as a suffix, but this is not necessary at all. ProANSI recognizes the files it saves[2].

- `Screen Mode...`

  This feature is not implemented in this version of ProANSI, but was left in because in the future it probably will.

- `Palette...`

  Allows you to change the on screen colors. Be careful to stick to the original ansi colors unless you know what you're doing. This option was implemented so you can adjust the colors slightly to adapt them to your monitor.

- `Font...`

  Here you can select a new font. Be careful to only use 8 by 8 fixed-width fonts. Especially under 1.x (when the "req.library" is used) you can select lots of other fonts, but this will not look good.

[1] When started from a Shell be careful with including ProANSI in your path, because if you do, you'll be able to start it from any directory and it probably won't find it's preferences.

[2] For the programmers amongst you, the format of the preferences file is supplied in the 'Technical Reference' chapter.

- `Keyblocks...`

  Opens the keyblocks editor, where all ten keyblocks can be modified. I think the window that opens kind of explains itself. You can select a keyblock by number and modify the keys by selecting them and changing their ascii code. Remember that any code can be selected here, but not all codes are printable in most programs.

- `Insert`

  Toggles the insert mode, which only works in normal mode (when `Anim Mode` is turned off).

- `Line Wrap`

  Toggles the line wrap mode, which affects the way the cursor reacts when it reaches the left or right border. This only applies to editing, and has nothing to do with the way screens are loaded or saved.

- `Page Wrap`

  Toggles the page wrap mode, which affects the way the cursor reacts when it reaches the top or bottom of the screen.

- `Destructive BS`

  Toggles the backspace key, which can be made destructive or not. If it is destructive, it will erase the character when you press the key, if not, the cursor will simply move left one character.

- `Anim Mode`

  Toggles the animation mode. The current project will be converted, but some information will be lost when going out of the animation mode, which is inevitable.

- `Save CR+LF`

  Toggles the saving of a CR (Carriage Return) with each line feed. This is necessary on most MS-DOS machines, and saves you the trouble of having to do this yourself. The save routines will take this into account when optimizing the length of the file, so enabling this option will be shorter than simply adding a CR in front of each LF by hand.

- `Save Clearscreen`

  Toggles the saving of a clearscreen code at the start of each file.

Chapter 4

ARexx Support

Probably the most powerful feature of ProANSI is its ARexx interface. This is the main reason why I started programming ProANSI in the first place. For example, how many times have you tortured yourself by loading and saving dozens of files, just because you want to create ascii versions of your ansi screens? Now you can do this with a simple script file. If you are SysOp of a BBS then you could also generate screens with arexx commands automatically, for example to generate new 'lastcallers' screens, or messages to next caller, or news bulletins, or... The possibilities are endless.

I've included lots of example scripts, including an AutoMenu script, which creates menus automatically from simple script files, in which you only need to specify the commands, (hot)keys and titles of each menu.

4.1 Using ARexx

For a more complete description of ARexx, you'll have to dig up your manual. I'll only explain the necessary things. ProANSI's message port name is "REXX_ProANSI" (mind the case!), and all commands are explained in the 'Command Reference' section. Please don't forget to turn on 'OPTIONS RESULTS' if you want to have results. ARexx is a language which can best be learned by using it, so have fun!

4.2 Command Reference

In general, all commands will return a return code of 10 when something went wrong, and 0 when the command succeeded. Some commands return

5 if you're in the wrong mode (in Normal Mode, undo for example won't work). Arguments must be supplied unless they are in square brackets.

4.2.1 quit

Stops the program as soon as possible, without further confirmation. Make sure you've saved your work.

Example: quit

4.2.2 version

Returns the current (FULL) version string. Make sure to turn on 'OPTIONS RESULTS', because the string is returned in the variable 'result'.

Example: version

4.2.3 go_up *[nr]*

Moves the cursor up. If the optional argument is supplied, you can state how many lines the cursor will move up. If you don't supply an argument, the cursor will go up one position.

Example: go_up 5

4.2.4 go_down *[nr]*

Moves the cursor down. If the optional argument is supplied, you can state how many lines the cursor will move down. If you don't supply an argument, the cursor will go down one position.

Example: go_down

4.2.5 go_left *[nr]*

Moves the cursor left. If the optional argument is supplied, you can state how many lines the cursor will move left. If you don't supply an argument, the cursor will go left one position.

Example: go_left 30

4.2.6 go_right *[nr]*

Moves the cursor right. If the optional argument is supplied, you can state how many lines the cursor will move right. If you don't supply an argument, the cursor will go right one position.

Example: go_right 7

4.2.7 set_pos *xpos ypos*

Places the cursor at a certain position. The origin (0,0) lies at the top left of the screen.

Example: set_pos 2 4

4.2.8 print_string *string*

Prints a string, starting at the current cursor position. This string can contain only printable ascii characters, so you're not allowed to include ansi codes. The *string* argument doesn't has to be in quotes when it contains spaces, because everything behind the command is treated as the string. The sting is printed in the current text colors.

Example: print_string This is an example.

4.2.9 select_col *fore back*

Selects a new *fore* and *back* color. This is the same as selecting a new color in the status window. Color values range from 0 – 15 for the foreground and 0 – 7 for the background.

Example: select_col 15 3

4.2.10 select_fcol *fore*

Selects a new foreground color (0 – 15).

Example: select_fcol 6

4.2.11 select_bcol *back*

Selects a new background color (0 – 7).

Example: select_bcol 3

4.2.12 set_col

Changes the fore- and background colors under the cursor (or block) to the currenly selected colors, just like the menu command `Set -- Both`.

Example: set_col

4.2.13 set_fcol

Changes the foreground color under the cursor (or block) to the currenly selected foreground color, just like the menu command `Set -- Foreground`.

Example: set_fcol

4.2.14 set_bcol

Changes the background color under the cursor (or block) to the currenly selected background color, just like the menu command `Set -- Background`.

Example: set_bcol

4.2.15 undo

Undoes the last change, just like the menu command. Of course, this only works in Anim Mode.

Example: undo

4.2.16 redo

Redoes the last change, just like the menu command. Of course, this only works in Anim Mode.

Example: redo

4.2.17 zap_undo

Zap undoes the last change, just like the menu command. Of course, this only works in Anim Mode.

Example: zap_undo

4.2.18 undo_all

Undoes all changes, just like the menu command. Of course, this only works in Anim Mode.

Example: undo_all

4.2.19 redo_all

Redoes all changes, just like the menu command. Of course, this only works in Anim Mode.

Example: redo_all

4.2.20 kill_redos

Kills all characters in the redo buffer, just like the menu command. Of course, this only works in Anim Mode.

Example: kill_redos

4.2.21 new

Starts a new project, without asking for confirmation, so be careful and save your current project before calling this function.

Example: new

4.2.22 preview

Previews the current screen or animation, just like the menu option.

Example: preview

4.2.23 get_xpos

Returns the current x-position of the cursor (in the variable 'result').

Example: get_xpos

4.2.24 get_ypos

Returns the current y-position of the cursor (in the variable 'result').

Example: get_ypos

4.2.25 get_fcol

Returns the current foreground color of the character under the cursor (in the variable 'result').

Example: get_fcol

4.2.26 get_bcol

Returns the current background color of the character under the cursor (in the variable 'result').

Example: get_bcol

4.2.27 get_char

Returns the current ascii code of the character under the cursor (in the variable 'result').

Example: get_char

4.2.28 save_ascii *filename*

Saves the screen as an (extended) ascii file. You must supply a (path and) filename.

Example: save_ascii menus:main-user.txt

4.2.29 save_ansi *filename*

Saves the screen as an ansi file. You must supply a (path and) filename.
Example: save_ansi menus:main-user.gr1

4.2.30 save_ansi_anim *filename*

Saves the screen as an ansi animation file. You must supply a (path and) filename.
Example: save_ansi_anim menus:main-user.anim

4.2.31 load_file *filename*

Opens a project from *filename*. It erases anything you're currently working on, so be careful.
Example: load_file menus:lastcaller.gr1

4.2.32 include_file *filename*

Includes *filename* to the current project.
Example: include_file menus:header.gr1

4.2.33 mark_block *x1 y1 x2 y2*

Marks a block. This is the same as marking a block with the mouse, except that you don't see the block when you mark it like this. It doesn't matter which corners of the block you specify.
Example: mark_block 10 20 40 5

4.2.34 cut_block

Cuts the marked block out of the screen, unless we're in Anim Mode.
Example: cut_block

4.2.35 copy_block

Copies the marked block, unless we're in Anim Mode.
Example: copy_block

4.2.36 paste_block

Pastes the block which was in the copy buffer back on screen, unless we're in Anim Mode. The topleft corner is the current cursor position, so you might want to set that first.

Example: paste_block

4.2.37 box_block

Draws a box around the currenly marked block, unless we're in Anim Mode.

Example: box_block

4.2.38 cut_line

Cuts the line out of the screen, just like the menu option, unless we're in Anim Mode.

Example: cut_line

4.2.39 copy_line

Copies the line, just like the menu option, unless we're in Anim Mode.

Example: copy_line

4.2.40 paste_line

Pastes the line which was in the line buffer back on screen, just like the menu option, unless we're in Anim Mode.

Example: paste_line

4.2.41 delete_line

Deletes the line, just like the menu option, unless we're in Anim Mode.

Example: delete_line

Chapter 5

Technical Reference

5.1 Preference file format

The preference files start with a four character ID, which currently is 'PAN1' but will be changed whenever the file format changes. After the ID, the following structure is saved. All fields in the preferences files are READ ONLY, and only apply to the current ('PAN1') file format.

```
typedef struct program_prefs
{
     USHORT         num_of_colors;
     USHORT         palette[16];
     UBYTE          x_size, y_size;
     char           font_name[40];
     USHORT         flags;
     UBYTE          keyblocks[10][18];
};

/*
 * Flag definitions, as used in the USHORT flags field:
 */
#define F_INSERT        0x0001 /* Insert/Overstrike      */
#define F_LINEWRAP      0x0002 /* Line Wrap (LEFT/RIGHT) */
#define F_PAGEWRAP      0x0004 /* Page Wrap (UP/DOWN)    */
#define F_DESTRUCTIVEBS 0x0008 /* Destructive BackSpace  */
#define F_UNDOREDO      0x0010 /* Enable Anim Mode       */
#define F_SAVECRLF      0x0020 /* Save LF as CR+LF       */
#define F_CLEARSCREEN   0x0040 /* Save Clearscreen Code  */
```

```
#define F_NOFOREGROUND  0x0080 /* No Foreground          */
#define F_NOBACKGROUND  0x0100 /* No Background          */
```

Chapter 6

Thank You

In this chapter I want to thank everybody who helped me with the development of ProANSI. Without these people, ProANSI wouldn't have been the program it is today.

> First I want to thank Albert-Jan Brouwer for his support when I'd encountered some weird crashes. Since I didn't have a MMU, I couldn't use the "Enforcer", so he did this for me. In the end, we discovered the bug, but without his help I would probably never have found it.
>
> Secondly I want to thank all the beta testers, who came up with some nice ideas and lots of bug reports. I want to thank Leo Elsinga for coming up with the name for my editor. Thanks also go to: Cor Knijnenburg, Aad Nieuwmans, Louis Roggeveen, Rene Te Pas, René Kuipers, Ed Mirck, Peter Kist, Peter Hagen and Frank Groen.
>
> Finally I want to thank Bard Papegaaij for learning me how to program in 'C' and especially how to handle large projects. And, last but not least, Joost Boerhout for sending me the documents of the ANSI/VT100 standards.
>
> Of course, there will be many people I've forgotten, so please don't be angry if your name isn't here. Just contact me and I will fix it!