

07a90c60-0

Hendrik De Vloed

COLLABORATORS

	<i>TITLE :</i> 07a90c60-0		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY	Hendrik De Vloed	July 25, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	07a90c60-0	1
1.1	LCDaemon Documentation	1
1.2	Introduction	1
1.3	Compatible displays	2
1.4	Requirements	2
1.5	The hardware	2
1.6	The software	5
1.7	Public Port commands	7
1.8	AREXX Port commands	11
1.9	Building the display	11
1.10	Writing display programs	12
1.11	How to get further development done	13

Chapter 1

07a90c60-0

1.1 LCDaemon Documentation

LCDaemon © 1995 VOMIT,inc.

Introduction What is all this about?

Requirements What do you need to use a LCDaemon-equipped Amiga.

The hardware "Technical" overview of the hardware part.

The software How to get any response from the hardware?

Hardware construction Now build it! (see mumbo-jumbo)

Software development How to write your own programs for LCDaemon?

Further developments & fun The funny part, and the thank-yous.

History So it wasn't bugfree from the start ???

" LINK "Why buy Windows 95 ? Why can't I just swap

" LINK my computer for a slower one?"

1.2 Introduction

LCDaemon is a combined hardware/software project to equip your Amiga with a Liquid Crystal Display.

Using this display, asynchronous messages can be displayed conveying all kinds of information to the user:

- System load
- Alerts
- Available memory
- Available resources
- Task-specific messages

The hardware consists of nothing more than a pre-assembled LCD module, available from the local electronic component store.

The software consists of a daemon-like program, presenting a public port to the system.

The total cost for the hardware should be around £20 - £40, depending on what type of display you purchase. Compatible displays should have the following **characteristics**:

- Hitachi LCD controller chip: HD44780A00 or compatible.
-

- 1 or 2 display lines.
- A character size of 5x7 pixels (5x10 possible, but futile)
- A line length of either 16 or 40.
- With or without backlighting.

A representation of the LCD is available: [click here](#). The picture is nearly scale 1:1: actual dimensions are 8.4cm x 4.4cm with a depth of max. 1.6cm. (1cm=0.3937in)

1.3 Compatible displays

The display used for the development was the Sharp LM16A21. This is a 16-character 2-line display without backlighting. The price of this display fluctuated from roughly £20 to £30, depending on the store.

When you purchase your display, chances are high you'll end up with a Sharp display, since this is a world leader in the manufacture of LCDs. Available displays include 16x1, 16x2, 40x1 and 40x2, all available with or without backlighting.

1.4 Requirements

To use this project, you will need:

- On the hardware front:
 - A **compatible** LCD display.
 - A male 25-pin D-connector for the parallel port.
 - A piece of 14-wire flatcable
- On the software front:
 - This package (preferably registered ;-)
 - Kickstart 2.0 or higher

1.5 The hardware

The principle of LCD displays

The modus operandi of LCDs is based on the polarisation of light and the polarisation-altering effects of certain substances.

Ambient light consists of electromagnetic waves.

These waves can be considered to possess a "wave direction", called polarisation. Waves can have any polarisation direction perpendicular to their direction. A three-dimensional figure illustrates this.

There are certain substances that can filter out a certain wave direction.

These substances are called polarisators or polarisation filters.

Polaroid sunglasses eliminate reflection by filtering out the main polarisation component present in light reflected on surfaces. [Click here to view](#)

a symbolic representation of the effect of a polarisation filter (the vertical lines on the schematic).

The 'liquid crystal' substances in an LCD have the following properties:

- At ease, the molecules are ordered horizontally, and have no effect on passing light.

- When brought into an electric field, the molecules turn gradually.

They can rotate the polarisation of passing light.

Using these properties, an LCD is constructed by putting the LC material inbetween two conducting layers and applying a voltage. The chemical properties of LC dictate that only AC voltage is acceptable.

On this figure you can observe a simplified representation of the functioning of an LCD. Note that the colours on the figure have nothing to do with the colour of the light, but are a symbol for the polarisation of the light, ranging from red (0°) over yellow (45°) to green (90°) and back to red (180°) etc...

Ambient light (white bar on figure) is fed through a polarisation filter.

The angle of this filter is referenced as 0° . Only light with this polarisation (actually all light, but with its intensity limited on the 0° -axis) is allowed to pass through the filter. In the right (inactive) pixel, the light just bounces off the mirror and passes unaltered through the filter, to your eye.

Resting, the LCD is clear (apart from the reduction in intensity by filtering out half the available light). When a tension is applied, the LC material is twisting the polarisation direction of the light. When the light has reached the filter after bouncing off the mirror, and the polarisation has turned 90° , all light is filtered out. You see a dark spot.

Since the amount the LC turns is dependent on the applied voltage, this can be utilized to adjust the contrast. To get a dark patch, the light must turn exactly 90° over the distance travelled in the LC.

It must be clear by now that the generation of the LC voltages is a matter of altering the phase difference of V_{bpl} and V_{seg} .

Unfortunately, on matrix displays there are far too many pixels to drive them one at a time. A complex 5-level block wave is used while the rows or columns are scanned one at a time. The waveforms involved become far too complex to generate conveniently using little electronics.

This is where chip manufacturers come in. Hitachi has developed a line of chips to produce exactly these waveforms and have conveniently interfaced them to the outside world using a 8 or $2*4$ bit command set.

These are the lines available to the developer:

- V_{ss} and V_{dd} : the power supply to the LCD and controller.

Since the chips are constructed using CMOS technology, the power drain is extremely low. In CMOS technology, the power dissipation is dependent on the switching frequency since 'all' power loss occurs when both p and

n channel transistors conduct. This occurs when changing logical state.

The Hitachi controller works at a frequency of around 250kHz, consuming less than 10 mW. A trifle compared to the dissipation of certain modern cooking ovens and stoves, like the Pain-tium or MC680[4]6]0.

A 1988 data sheet for my display reveals a maximum power dissipation of 2.2 mA at 5V. However, a measurement of the total current drawn by the LCD (measured in the return feed on the Vss pin for convenience purposes) produces only 0.35 (rest) to 0.38 (a screen full of symbols) mA. Hitachi must have improved their technology since then. As the other inputs are CMOS, no significant power drain should be expected.

- Vo: the contrast adjustment line.

The operating instructions suggest connecting a trimmer (adjustable resistor) of 10kOhm between Vdd and Vss and connecting Vo to the middle of the trimmer. However, the data sheet reads a typical voltage of 0.65 V for Vo. Since a trimmer of 10kOhm would draw 0.5 mA this seems silly.

In [this design](#), Vo has been connected to ground. This produces a contrastful display, but involves no readability problems. If your display is badly readable, do this.

- RS: Register Select.

This line determines whether the data represents actual character codes (RS=1) or control information (RS=0).

- R/W: Read/not(Write).

When 0, the display listens to data on the data bus. When 1, the display will put data on the bus.

Since the LCD will be interfaced to the parallel port, this line is permanently connected to 0 for safety reasons. If R/nW was controllable by software and an unexpected situation would occur (e.g. starting a program that fiddles with the lines on the parallel port connected to R/W and E) and the parallel port data buffers were set for output, this could lead in both the LCD and the Amiga CIA chip trying to put different data on the same lines.

Since the CIA has active pull-ups, there is no safe way of interfacing the LCD to the parallel port in a bidirectional fashion.

The software will deal with this 'problem'.

- E: Enable.

A pulse on the E line activates the controller.

The controller will pick up data off the bus (or put it on the bus, according to R/nW).

- DB0 ... DB7: the data bus.

This should need no further clarification.

1.6 The software

In order to drive the hardware a suitable driver is needed. Certain electronics magazines have published a demonstration program for the PC¹, allowing the user to select options from a menu which perform various actions for the user. It was written in GW-Basic, and relied on its slowness for certain timing issues.

Pathetic! :-)

The Amiga approach has to do better than that. It must really open up the LCD to all other tasks. Therefore, the LCD program has the following interesting aspects:

- A **public message port** to which tasks can send messages.

The messages include priority and time controls, allowing arbitration to select which messages make it to the LCD in heavy-traffic conditions.

- An **AREXX port**, allowing programs that are not specifically written for the LCD to benefit from an extra display device.

- Example drivers are included, allowing the LCD to be driven from virtually anything (provided you write the port driver)

IMHO, the LCDaemon program included lives up to these expectations.

LCDaemon needs some parameters at startup. These are described below.

As always, the parameter list can be asked by typing a ? as only parameter on the command line.

LINEs or **HEIGHT** Set the physical parameters of your LCD. Since there **CHARACTERs** or **WIDTH** is no way of finding out these characteristics from (keyword required) the display itself you need to set these at startup.

The defaults correspond to my Prototype LCD:

example: **CHARACTERs** 16 **LINEs** 2

TASKPRI or **TP** Change the multitasking-priority of the daemon to increase responsiveness. Defaults to 0.

DESCENDERs or **D** If set, characters like j,q,p and so on will use the bottom line of pixels. Some LCD's have a little gap

between this line and the rest of the matrix, so

you may get ugly-looking results when using this switch.

Defaults to no descenders.

CURSOR or **C** Activate a blinking cursor on screen. Purely

(switch) aesthetic. Defaults to my aesthetic level

(no cursor) :-)

STARTUP This field contains a driver-dependent string that

is passed to the port-specific driver. For example, the `lcd_ami` Amiga window driver uses this field to set the window's title. The `lcd_pit` driver uses this field to select the port number (0 or 1).

Since this field is set only once by the user, no restrictions on the format of parameters you want to define are present. Just document this field in your driver info. If spaces are present, enclose this field in " characters.

There are currently four drivers available:

- `lcd` The standard parallel port driver. `Parallel.resource` is locked so no simultaneous printer access is available.
- `lcd_pit` The Multifacecard 2 PI/T driver. The Motorola Parallel Interface/Timer chip is accessed directly, so it is your responsibility to ensure the `pit.device` is not used. I have noticed the `pit.device` is rather aggressive in writing the control registers. When accessing the PI/T chip, the bits belonging to the other port in a shared register are rewritten. This driver tries to keep the right values on the right bits, but some LCD interference when printing to the other port is possible. This driver is programmed properly: no interference on the other port is noticeable when LCD data is sent.

To select which port to use, you can use the "startup" parameter, followed by the number of the port (0 or 1).

- `lcd_mfc3` The MultifaceCard 3 driver. This has been tested by one user and has been reported to work; Note, however, that the pin-connection on the sub-D connector resembles the PC¹ standard. Pin 14 contains the Centronics standard AUTOFEED signal instead of the +5V power line that is common to the Amiga parallel port pin. In this driver, the AUTOFEED signal is continuously held to a logic '1'. But the Motorola documentation only specifies a corresponding voltage of 2.4 V min. My beta-tester reports a voltage of 4.5 V in his case. Your mileage may vary. Notice, however, that all current goes through the Motorola Parallel Interface Adaptor chip and I won't be held responsible if you blow it. Recommended action

is to connect the Vdd pin of your LCD to your internal power supply. (If you have a MFC3 card, you have a desktop Amiga anyway).

· `lcd_ami` This driver emulates the LCD in a window on your Amiga workbench. This way, you can have a look at this software without actually building the LCD. To use this driver properly you need Kickstart 3.0 (for its pen allocation features) and a workbench with sufficient colours to be able to allocate dark and light green. If your display remains blank, try increasing the number of colours of your WB screen. There is a little bug that I haven't bothered to fix yet: a small garbage sign is visible after the text. As I mentioned before, you will probably start this driver once in a lifetime so I can't be bothered to fix this. Unless you really complain about it to me. :-)

¹ Arrogant abbreviation for IBM-compatible Personal Computer

1.7 Public Port commands

The public message port "«« LCD rendezvous »»"

In `lcd.h` programmers will find all necessary information to drive the LCD through use of the public inter-task message port instead of sending AREXX messages. Not only does this relieve REXX of processing all those silly string-oriented messages, but it allows finer control of the LCD.

Doing something to the LCD involves:

- Allocating the needed structures to allow inter-task communications.

(Allocating a replyport, a `lcdmessage` structure, ...)

- Filling the `lcdmessage.lcd_Code` field with the command type:

- `LCDMSG_ASKPARAMS`:

`lcdmessage.lcd_Data` contains a pointer to a `lcdparams` structure. After successful completion its fields will contain data for the LCD screen used. This data is supplied by the user on the command line while starting the daemon. This command is implemented to free the user of having to set his/her configuration in each program that uses the LCD.

· LCDMSG_QUEUE:

The `lcdmessage.lcd_Data` contains a STRPTR to a string that has to be displayed. To allow the driver to select which messages are important enough to display the `lcdmessage.lcd_Priority` field will determine how important a message is. Use the supplied LCDPRI_ defines to indicate your type of application. The user should be given an additional parameter that can be added (positively or negatively) to this amount. The user-allowed range should be -10...10, thereby allowing "depth arrangement" of various tasks of the same LCDPRI_ type.

The `lcdmessage.lcd_Ticks` is the time in 50ths of a second the message has a chance (according to the priority) of being displayed.

· LCDMSG_ALLOCATELCD and LCDMSG_FREELCD:

Allow a program to gain exclusive access to the LCD.

From now on the LCD will only respond to LCDMSG_LCDIRECT commands. Always remember to check successful completion of an ALLOCATE before assuming your LCDIRECT messages will be obeyed!

· LCDMSG_LCDIRECT:

The `lcdmessage.lcd_Data` field contains a pointer to a `lcdirect` structure. This structure consists of two LONGs: a command longword that has to be filled with one of the LCDCMD_ commands. These symbolize the basic commands the LCD responds to:

- × LCDCMD_CLS : clear LCD
 - × LCDCMD_HOME : return to (0,0)
 - × LCDCMD_CURSDIR: `lcdirect.Data` has to contain either RIGHT or LEFT, which is the direction the cursor will walk on successive data writes.
 - × LCDCMD_DISPLAY: `lcdirect.Data` = ON or OFF.
 - × LCDCMD_CURSOR: `lcdirect.Data` = ON or OFF. Controls the visibility of the cursor.
 - × LCDCMD_BLINK: `lcdirect.Data` = ON or OFF.
 - × LCDCMD_SHIFTD: `lcdirect.Data` = RIGHT or LEFT. Shifts the entire display.
 - × LCDCMD_SHIFTC: `lcdirect.Data` = RIGHT or LEFT. Shifts the cursor.
-

× `LDCMD_GOTOXY`: `lcdirect.Data = (new_y_pos<<16)|(new_x_pos)`

× `LDCMD_GETXY`: `lcdirect.Data` contains a pointer to a `LONG` that will be filled according to the formula in `GOTOXY`.

× `LDCMD_DATA`: The data in `lcdirect.Data` is written to the LCD display or `CGRAM`¹ memory.

· `LCDMSG_ALLOCATEHANDLE` (1.6)

When this command returns without `lcdmessage.lcd_Error` the `lcdmessage.lcd_Data` field points to a struct `lcdscreen` (defined in `lcd.h`). The `lcdscreen` structure contains the following fields:

× `lcdscreen.version` is initialized to `LCDaemon`'s version of the `LCDScreen_MINVERSION` constant. Check for this version against the `LCDScreen_MINVERSION` of your header file and fail if it is smaller.

× `lcdscreen.bufferwidth` and `lcdscreen.bufferheight` are initialized to reflect the screen parameters. Note that the `bufferwidth` variable corresponds to the width of one line of characters in the buffer, but does not necessarily correspond to the width of the display. Use the `LCDMSG_ASKPARAMS lcdparams.width` field if you need the actual width.

× `lcdscreen.screenbuffer` points to a buffer of `lcdscreen.bufferwidth * lcdscreen.bufferheight` bytes.

This buffer is yours to write and is double buffered in `LCDaemon`, so you needn't worry about modifying this buffer during its appearance on the LCD.

These fields are private, and must remain static during the lifetime of the handle. Neither you nor `LCDaemon` are allowed to modify them.

· `LCDMSG_UPDATEHANDLE` (1.6)

The `lcdmessage.lcd_Data` must point to the `lcdscreen` structure, as returned by `LCDMSG_ALLOCATEHANDLE`.

Once a handle has been established, you are free to modify the following fields:

× `lcdscreen.customcharheight` indicates the number of bytes per character for the user-programmable characters. This should be set to 8, since the currently supported LCDs use this amount.

× `lcdscreen.customcharnum` contains the number of programmable characters, each of height `lcdscreen.customcharheight`.

The number of custom characters can be chosen arbitrarily, but note that the currently supported LCDs have a limit of 8 custom characters. The current screen updating algorithm dumps the entire user-character set into the LCD at each context change where different handles with their own programmable characters are involved, so it is useful to limit this number to the number of actually used programmable characters.

× `lcdscreen.ud_flags` must be set to reflect to changes in the above parameters you made since the last `LCDMSG_UPDATEHANDLE`.

The value for this field is a logic OR of the flags:

» `LCDUPD_DISPLAY` : The contents of the screen buffer have been altered and need to be updated.

Updating is done using a smart algorithm to avoid unnecessary writes to the LCD.

» `LCDUPD_CUSTOMCHARNUM` : The number of custom characters needs to be altered. Currently, only a `lcdscreen.customcharnum` change is allowed, not a `lcdscreen.customcharheight` change.

» `LCDUPD_CUSTOMCHARDEFS` : Either the custom char data or the pointer `lcdscreen.customchardefs` has changed.

· `LCDMSG_FREEHANDLE` (1.6)

This deallocates all storage allocated by the `LCDMSG_ALLOCATEHANDLE` call. The `lcdmessage.lcd_Data` must contain the pointer to the `lcdscreen` structure, as returned by the `_ALLOCATE` call. The screen buffer is deallocated as well, since it was allocated by `LCDaemon`. The custom char data however, is the property of the application and will not be deallocated.

¹ Currently, the Character Generator RAM can not be programmed by `LCDCMD_DIRECT` commands. Custom character support is implemented using `LCDMSG_ALLOCATEHANDLE` and related commands. Contact me if you really need this feature.

1.8 AREXX Port commands

The Arexx message port "LCDAEMON"

This port currently supports three commands:

Parameters in [square brackets] are optional.

- lcdmessage [time t] [pri p] text

An 'echo' equivalent. Maps to the **LCDMSG_QUEUE** command on the **public port**.

- getcharacters

- getlines

After completion, the Result variable contains the parameters of the LCD at startup. Similar to **LCDMSG_ASKPARAMS**.

Do not forget to issue the OPTIONS RESULTS Arexx command to allow the Result variable to be filled!

1.9 Building the display

The author takes no responsibility whatsoever for what could happen to you, your computer, the LCD or anything connected to your computer by any physical (liquid, gaseous, solid, energy) connection, even if this document contains blatant lies.

This means, if you blow up your local nuclear plant after printing the word "Tschernobyl" on the LCD, or your cat drops dead after watching a little pixel mouse dance and make obscene gestures¹,

I am not to blame. The only insurance this document is not a really great April 1 joke intended to increase the Comm... er Escom... er Viscorp Amiga sale and 2nd-hand fried part market is the current date.

Make sure you get a data sheet for your LCD and verify these pinouts. Furthermore, check you have tied these to the correct parallel port pins before plugging it in! (see p. 319 Hardware Reference Manual 3rd Ed.)

The Sharp LM16X21A and compatibles have the following pin layout:

1 Vss : Connect to GND of parallel port (pins 18-22)

2 Vdd : Connect to +5V (pin 14 on Amiga parallel ports)²

3 Vo: Contrast. Either use an adjustable resistor or tie to GND.

4 RS: Register Select: tie to POUT (pin 12)

5 R/W: Read/notWrite: ALWAYS tie to GND!!! If you read from the LCD the LCD will try to put data on the parallel port

while the port is configured as output! This will probably result in blowing your LCD (or even your CIA)³.

6 E: Enable: tie to BUSY (pin 11).

7 \

. \

. \}DB: Data bus: tie to data lines of parallel port (pins 2 to 9).

. /

14/

If you have a backlit LCD you'll have to connect the Vled and Vss lines to a separate² 5 V power supply.

¹ Possible as of version 1.6.

² Make sure that you don't draw too much current.

The port should be able to handle the 1 to 10 mA required for the LCD. When you use a backlit LCD, the hundreds of mA to illuminate it will have to be supplied by some other means! ³ See a picture of the problem.

The LCD is a fragile CMOS IC so it will probably blow first and act as a very expensive fuse :). For more info, read [here](#).

1.10 Writing display programs

All programmer information can be found in the [Public](#) and [Arexx](#) sections.

Please keep in mind that your programs will always be struggling to get information on the LCD since they will be competing for time.

Be sure to check return codes (lcd_Error field) of your messages.

Certain error messages might need some further clarification:

- LCDERR_UNKNOWN: The lcd_Code is unknown. Nothing has happened. Maybe you could prompt the user to upgrade to a more recent driver that supports the violating LCDCMD command if you really need it.

- LCDERR_TOOBUSY: This can occur in various situations, for example when the message could not be queued because the LCDaemon is shutting down or when the LCD is allocated for exclusive access by another program (LCDMSG_LCDIRECT command LCDCMD_ALLOCATE)

- LCDERR_YOURFAULT: You forgot to initialize fields or initialized them to gibberish values. Naughty boy!

The LCDMSG_LCDIRECT commands are not user-protected. You could

issue direct commands even if the lcd is allocated by another task.

The purpose of the LCDMSG_ALLOCATE command is not to fight off tasks, but to politely ask for exclusive access. When this request is denied your courtesy is counted on to refrain from sending LCDMSG_LCDIRECT commands. If you don't, chances are 100% you'll end up with on-screen gibberish, since the messages of the offending task will be intertwined with the messages of the task that properly allocated the LCD. So:

Please please please CHECK and RESPECT return codes!

As always for a public message port, send messages to it through the SafePutToPort() function, which disables multitasking, checks for the existence of the port, sends the message and enables multitasking.

This ensures that the port didn't close under your very nose while you were about to send a message to it. This function can be found in the example programs

For ARexx programs: don't forget to use the command

OPTIONS RESULTS

before using commands that are supposed to return something, like `getcharacters` or `getline`!

1.11 How to get further development done

Please read this! It's not boring, like all those other disclaimers!

I'm not begging for money like all other PD authors! ;-)

Nowadays, most software written seems to bear the shareware label.

No matter how insignificant or futile the utility, the user is given a moral obligation to pay a certain amount. In the end, this means that --- to be at "moral ease" --- the average user should fork out hundreds of pounds before a decent working environment can be obtained. Fortunately, most shareware programs are fully functional. Combined with the fact that the average user doesn't really have an urge to pay money this has led to horrible contraptions like "crippleware".

Several Amiga users --- including myself --- have adopted a "What? Crippleware? del¹ *.lha!"-attitude towards this pathetic attempt to abuse Aminet and other data carriers for free publicity. For this is exactly what crippleware is: a program, totally made useless on purpose, is put on Aminet in the hope that the user will pay the amount to buy it in order to get to work with it. No "if you use it intensely" underlying idea like shareware, just abuse of Aminet to make publicity.

What's worse: some shareware authors have descended to using "threat":

I have heard that in my (registered) copy of Spot there is embedded program code to do harm if a tampered keyfile is detected. Have they gone totally mad??? I fear for the day that Reorg messes up my personal keyfile, thereby triggering Spot into who-knows-what frenzy !

In ArcHandler's guide I read a similar thing: "del¹ ArcHandler.lha" was my spontaneous reaction. In one part of the doc file the authors dismiss all responsibilities for misbehaviour of their programs, in the other they state without remorse their programs have been programmed to do nasty things. Do I qualify for Safe Hex International's "Give us names of virus programmers" reward? ;-)

One sad day some program's history file will contain the following lines:

Revision 3 : Fixed small bug in ARexx port code

Fixed accidental formatting of HD due to bug in
keyfile protection code. (sorry, registered users!)

Fixed windo^%\$#° Ð©¥°®

<file ends due to author killed in stampede of enraged ß-testers>

I bought Meeting Pearls III, and found a crippleware port of the shareware UN?X program XV. Huh? Asking money (\$10 for ONE update) for porting someone else's shareware? What are we supposed to do? Register TWICE??? I sure hope you don't get into trouble with the original author, Terje!

The combination of these ideas has led to the fact that LCDaemon is...

...

...

... freeware! (hurrah! ;-))

That doesn't mean you can include this into other products, though!

If you want to use this commercially, contact me.

If you want to use this project in a magazine or on a coverdisk, the "price" is 1 year's subscription starting from the issue the project appears in.

CD-ROM distribution is only allowed on CD-ROMs that satisfy 1 or more of the following:

- The name is "Aminet" followed by a number (Thanks, Urban, great job!)
- The name contains "Fish" (great job, Fred!)
- The CD sells for <= £15, DM25 or equivalent.

This archive may be carried anywhere as long as no alteration is made, not even adding file_id.diz or other junk.

A PD disk with this archive cannot be sold for more than £1 or equivalent.

And now something for YOU:

The only thing you, average fanatic Amiga user, could do for me is sending me a postcard or Email stating why you (dis)like the program, what could be done or what bugs you found.

If you don't have the courage to get out of bed and drop a piece of paper in the mailbox, consider this program my tiny attempt to reward all other Amiga PD authors for their excellent work in thwarting the Gates of Hell.

Hendrik De Vloed

VOMIT, inc.

Pontstraat 22

B-9090 Melle

Belgium, man, Belgium!²

E-mail: hendrik.devloed@barco.com

Thanks

So far, I have received E-mail from about twenty people who are interested in building a display or have already built it.

Thanks to Rob Darke in particular, for uploading the original archive.

Please get in touch if you want features! The programmable character support wouldn't have been implemented if I didn't know someone was waiting for it! Since the current driver fulfills all my needs, it is now up to you to give me input for the features you want!

¹ I use CShell :)

² © Z. Beeblebrox