# AmigaLoad

Hans Forssell

**COLLABORATORS**

| | *TITLE* :<br><br>AmigaLoad | |
|---|---|---|

| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
|---|---|---|---|
| WRITTEN BY | Hans Forssell | February 11, 2022 | |

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|---|---|---|---|
| | | | |

# Contents

# Chapter 1

# AmigaLoad

## 1.1 AmigaLoad V2.0

```
  /\                __          |    _         _
 /  \   /\/\   |  /  _  /\   |   / \  /\  | \
/----\ /    \  |  \__/ /--\  |__ \_/ /--\ |_/  V2.0
```



Written by Hans Forssell Email: tl94hfl@student.hgs.se
Stigsrundan 13
806 42 Gefle
Sweden

## 1.2 What is AmigaLoad

AmigaLoad is software and hardware that makes it possible to display some information (CPU-load, Free mem, ...) about your Amiga on a Moving Coil Instrument (MCI), LED-display, LCD or similar.

AmigaLoad can also display the information on your WB screen.

Don't be afraid to send questions, suggestions and bug-reports to me! I will try to answer all your mail (tl94hfl@student.hgs.se).

```
You can always download the latest version of AmigaLoad from my
HomePage: www.hgs.se/~tl94hfl/
```

## 1.3 What is new?

```
Thank you for all suggestions! I have not implemented all in this
release, but maybe in the future.

News in AmigaLoad 2.0
   Support for LCD-display hardware
      (Thanks to Hendrik De Vloed for LCDaemon!)
   Different virtual instrument types at the same time
      (Suggested by Haidinger Walter)
   Icons updated
      (Many thanks to Len Trigg for the icons!)
   New CPULoad
      (Suggested by Haidinger Walter)
   Possible to have two rows/columns even if odd number of instruments
   Limited ARexx support
   68020+ version
   Snapshot added in AmigaLoad menu
   Possible to change colors and inner spacing for virtual instruments
   Possible to Move/Resize windows that has no Dragbar/Size gadget
   Bugs Removed:
      Many groups had illegal colors

News in AmigaLoad 1.3
   Support for LED-display hardware
      (Many thanks to Yannick Erb for the LED hardware!)
   New icons
      (Many thanks to Len Trigg for the new icons!)
   Possible to add/remove window border, dragbar and size gadgets
      (Suggested by Haidinger Walter and Torbjorn Aronsson)
   Virtual window can be of backdrop type
   Possible to have the virtual instruments in two rows or two columns
      (Suggested by ???. Sorry I have forgot your name!)
   Two new types of the 'Graph virtual instrument' added
      'Graph (Filled)' and 'Graph (Scale + Filled)'
      (Suggested by Len Trigg)
   Time added (Warning: The analog type looks terrible...)
   Possible to watch free space on any device.
      (Suggested by Sven Differt)
   Possible to watch stack usage for any task.
   AmigaLoadNoGraph for those who don't use the 'virtual instruments'.
      (Suggested by Sadik Hafizovic)
   Bugs removed:
      AmigaLoad window activated when opened
         (Found by Haidinger Walter)

News in AmigaLoad 1.2
   Possible to invert all values
   CPULoad rewritten
   Three new virtual instruments
      Two resizeble MCIs
         (Suggested by Fredrico Villata, Alexander Reifinger, ...)
```

```
        Graph with scale)
          (Suggested by Haidinger Walter)
   Uptime added
      (Suggested by Haidinger Walter)
   Many bugs removed:
      System freezed if CPU-caches were used on some systems
         (Many thanks to Shawn D'Alimonte for a lot of testing!)
      CTRL_C handling bug removed
         (Found by Haidinger Walter)
      Quit problems when priority < 0 removed
         (Found by Haidinger Walter)

News in AmigaLoad 1.11

   Possible to change the order of the virtual instruments
      (Suggested by Michele Stival)
   Two new virtual instruments (Gauge and Gauge with scale)
   Many bugs removed:
      System freezed if CPU-caches were used on some V37 systems
         (Found by Shawn D'Alimonte and Peter Daas)
      Default path to AmigaLoadSettings changed to Sys:...
         (Found by Pater Daas)
      Now Public Mem works!?
         (Found by me)

News in AmigaLoad 1.1

   Support for two MCI:s                          (Suggested by David Bump)
   Two new 'Types' (Frag Fast and Frag Chip)
   Two new virtual instruments (Numeric and Graph)
   Set the path to AmigaLoadSettings
   Bug removed in 'Calibration'
   The MCI routines are a bit faster than before
```

## 1.4 Copyright

```
AmigaLoad 2.0 is written by Hans Forssell 1997

I take no responsibility if you damage your computer hard and/or
software when you use this program.

AmigaLoad 2.0 is FreeWare.
```

## 1.5 Installation

```
AmigaLoad 2.0 requires WB2.0 and MUI 3.8.

Unpack AmigaLoad.lha somewhere on your HD.

If you have a 68020 or better:
   Delete AmigaLoad and AmigaLoadSettings.
   Rename AmigaLoad020 to AmigaLoad.
```

Rename AmigaLoadSettings to AmigaLoadSettings.

If you have a 68020:
Delete AmigaLoad020 and AmigaLoadSettings020 to save diskspace.

Copy AmigaLoad to the WBStartup drawer, and AmigaLoadSettings to Sys:Prefs/.

AmigaLoad 2.0 use some of your old AmigaLoad.prefs settings.

## 1.6 MUI

AmigaLoad 2.0 require MUI 3.8 or better.

This application uses

MUI - MagicUserInterface

(c) Copyright 1992-97 by Stefan Stuntz

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called "muiXXusr.lha" (XX means the latest version number) on your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send

DM 30.- or US$ 20.-

to

Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY

Support and online registration is available at

http://www.sasg.com/

If you for some reason not have installed MUI yet - do it now!!!

Thanks to Stefan Stuntz for a Magical User Interface.

## 1.7 AmigaLoad

Double click on the icon to start AmigaLoad. If you have a MCI/LED connected to the joystickport it will display 0% until you have run AmigaLoadSettings. AmigaLoad will also open a window on your WB screen that displays the CPULoad and Free mem.

AmigaLoadNoGUI is AmigaLoad without the GUI. You can still use ↩ AmigaLoadSettings.

If you for some reason want to quit AmigaLoad use Exchange or click on the close gadget in the 'Virtual' window.

AmigaLoad menu:

Settings MUI...
: Start MUIs settings program. The changes effects only AmigaLoad.

Settings...
: Start AmigaLoadSettings. See AmigaLoad Settings/Common/Settings for more info.

Snapshot
: Snapshot AmigaLoads window.

See Type for more info about the different display types.

## 1.8 AmigaLoadSettings

Double click on the icon to start AmigaLoadSettings.

The config is split up in seven pages:

| | | |
|---|---|---|
| Virtual | : | Settings for Virtual instruments |
| User | : | Settings for Virtual and MCI/LED |
| Invert | : | Settings for Virtual and MCI/LED |
| Common | : | Settings for Virtual and MCI/LED |
| LCD | : | Settings for LCD |
| MCI 1 | : | Settings for MCI 1/LED 1 |
| MCI 2 | : | Settings for MCI 2/LED 1 |

If you have one or more MCI:s connected read Calibrate MCI!
If you have one or more LED:s connected read Calibrate LED!

Save: Save and quit.

Cancel: Quit.

## 1.9 Virtual

The Virtual config is split up in four pages:

| | | |
|---|---|---|
| Display | : | Instruments |
| Window | : | Window size, position and type |
| Type | : | Instrument types |

```
 Setpen   : Colors and inner spacing
```

## 1.10 Display

```
Set which instruments that should be displayed in the
'Virtual' window on the WB screen.

For those of you who are not familiar with MUI:s drag and drop:
   The left list lists the virtual instruments that are not displayed,
   and the right list lists those that are displayed.

   To display a new instrument:
      Move the pointer to the instrument you want to display.
      Press the left mouse button, and drag the instrument to
      the right list.
      Release the button.

   To remove an instrument:
      Drag the instrument from the Display list to the Available list.

   Change the order of displayed instruments:
      Drag the instruments in the Display list. A horizontal line tells
      you were the instrument will be placed.

   See Type for more info.

Short explanation of the gadgets...

Type:
   Select one of AmigaLoads 11 different types of virtual instruments.
   See also Virtual/Types

Uptime
   With the cycle gadget you can select how the Uptime should
   be displayed. Try the different types to find the one that
   looks best with the selected type of virtual instrument.

   The horizontal format is:
      D-HH:MM

   D = Days
   H = Hours
   M = Minutes

Time
   With the cycle gadget you can select how the Time should
   be displayed. Try the different types to find the one that
   looks best with the selected type of virtual instrument.

   The digital format is:
      HH:MM
   H = Hours
   M = Minutes
```

## 1.11 Window

Set orientation, type and size of the 'Virtual' window on the WB screen.

Short explanation of the gadgets...

Direction
   Select if window should be horizontal/vertical or two rows/columns.
   Note: There must be an even number of objects to use two rows/columns

Window Type
   Select if the AmigaLoad window should be backdrop or not.

Border
   Add/Remove window border.

Dragbar
   Add/Remove window titlebar and size gadget.

SizeGadget
   Add/Remove window size gadget

MUI Snapshot
   Turn 'MUI Snapshot' ON or OFF.

   When ON, the window position and size is controlled by MUI.
   Use Snapshot in MUIs popup menu or Snapshot in AmigaLoads menu.

   When OFF, the window position and size is controlled by
   Left, Width, Top and Height string gadgets.

Left
Width
Top
Height
   Window position and size when 'MUI Snapshot' is OFF.

## 1.12 Type

Set the type of virtual instrument. This makes it possible to have
Graph for CPULoad and Gauge for memory.

If 'Default' the type in Virtual/Display/Type is used.

## 1.13 Setpen

The Setpen config is split up in three pages:

 MCI    : Colors and inner spacing for MCI
 Graph  : Colors and inner spacing for Graph
 Time   : Colors and inner spacing for Clock

## 1.14 MCI

```
Short explanation of the gadgets...

Type
   Select if MUIs default background or user selected background
   should be used.

Background
   Background color for MCI. (Normal MCI type.)

Scale
   Scale color for MCI.

Pointer
   Pointer color for MCI.

Left
Right
Top
Bottom
   Set inner spacing. Inner spacing is the number of pixels
   between the frame and the MCI.

   Note: The inner spacing is not updated automaticaly, you must
   press update!

Update:
   Update inner spacing.
```

## 1.15 Graph

```
Short explanation of the gadgets...

Type
   Smart Refresh
   Use this type with smart refresh windows. This can be used
   with simple refresh windows AmigaLoad is the frontmost window.
   This is the fastest type.

   Simple Refresh
      Use this type with simple refresh windows.

   MUI Background
      Use MUIs default background.

Background
   Background color for Graph. (Smart and Simple Graph refresh type.)

Scale
   Scale color for Graph.

Curve
   Curve color for Graph.
```

Left
Right
Top
Bottom
   Set inner spacing. Inner spacing is the number of pixels
   between the frame and the Graph.

   Note: The inner spacing is not updated automaticaly, you must
   press update!

Update:
   Update inner spacing.

## 1.16 Time

Short explanation of the gadgets...

Dial
   Dial color for analog clock.

Scale
   Scale color for analog clock.

Pointer
   Pointer color for analog clock.

Left
Right
Top
Bottom
   Set inner spacing. Inner spacing is the number of pixels
   between the frame and the MCI.

   Note: The inner spacing is not updated automaticaly, you must
   press update!

Update:
   Update inner spacing.

## 1.17 User

These settings are common for both virtual instruments, MCI/LED and LCD.

See Type for more info.

## 1.18 Invert

If set the value will be inverted (0% -> 100%, 100% -> 0%) for
virtual intstruments, LCD and MCI/LED.

## 1.19 Common

These settings are common for both virtual instruments, MCI/LED and LCD.

Short explanation of the gadgets...

Update: How often should the values be updated (MCI/LED, LCD and Virtual instruments) in seconds? Default = 3s.

Settings: Path to AmigaLoadSettings. This path is used when you select 'Settings...' from the AmigaLoad menu.

Code: Registration code. Send a mail to me with the subject 'AmigaLoad 2.0' and I will send the code to you. The registration is totaly free.

## 1.20 LCD

Settings for the LCD display.

LCD commands : Command info. Also available as bubble help.
LCD examples : Some examples

There are four LCD strings:
Start : Only displayed when AmigaLoad starts.
Init : Displayed when AmigaLoad starts, and when LCD settings have changed.
Main : Displayed for every update.
End : Only displayed when AmigaLoad ends.

Note : All text is first written to AmigaLoads internal buffer, and not directly to the LCD. The buffer is copied to the display at the end of the string or after a %! or %d command.

The Init string is for static data. For small LCD displays this can be moved to String almost without any speed loss.

Short explanation of the gadgets...

LCD:
Turn ON/OFF AmigaLoads LCD update. If you don't have a LCD connected set LCD to OFF to avoid annoying delay when AmigaLoad is started.

Start:
This string is written to the LCD when AmigaLoad starts.

Init:
This string is written to the LCD before the 'String' is written for the first time (After startup and changes in LCD settings).

Main:
This string is written to the LCD every Common/Update second.

End:

```
   This string is written to the LCD when AmigaLoad quits.

Type 1-3
   Used by LCD commands %s and %v.

Test Start:
   Writes the Start string to the LCD.

Test End:
   Writes the End string to the LCD.
```

## 1.21 LCD Commands

```
The '%' character is AmigaLoads command character. The '%' is followed by
one or more characters describing what AmigaLoad should do.

To write the '%' character to the LCD use %%.

The '_' character is ignored. Use this to separate commands. If you need the
'_' character you can define a custom character. See %z!

If AmigaLoad finds a illegal command it writes a '?' to the LCD. Example:
   Abc_%q_def
   %q is not a LCD command, and the output is:
   Abc?def

All examples assumes that:
   Type 1 = New CPULoad
   Type 2 = Ready
   Type 3 = Fast mem
   Type 4 = Chip mem

   Time: 21:40 Uptime: 1 day, 7 hours and 42 minutes.

   and a LCD-display that is 16*2 characters.

See also LCD examples!

Commands:

 Cursor       : (%h %x %y)    Cursor movement commands.
 Type         : (%s %l %r %v) Type 1-4 commands.
 Update       : (%! %d)       Update LCD display commands.
 Clear        : (%c)          Clear LCD commands.
 Text         : (%m)          Text commands.
 Time/Uptime  : (%t %u)       Time/Uptime commands.
 Gauge        : (%g)          Gauge commands.
 Special      : (%a %o %% %*) Special commands.
 Custom char  : (%zd %z)      Custom char commands. (Advanced)
```

## 1.22 Cursor

```
%h<n>    Home
   <n>   Row. Default 0.

   %h    Move cursor to (0, 0).
   %h1   Move cursor to (0, 1).

   See also %x and %y!

   Note: Upperleft corner is (0, 0)!

%x<n>    Move cursor to column <n>.
   <n>   0-99
         +    Next character.
         -    Previous character.
         @    Last visible character.

   %x10  Move cursor to column 10.
   %x+   Move cursor to next character.
   %x@   Move cursot to last visible character (15).

   Note: Upperleft corner is (0, 0)!

%y<n>    Move cursor to row <n>.
   <n>   0-99
         +    Next row.
         -    Previous row.
         @    Last visible row.

   %y1   Move cursor to row 1.
   %y+   Move cursor to next row.
   %y@   Move cursor to last visible row (1).

   Note: Upperleft corner is (0, 0)!
```

## 1.23 Custom char

```
%zd<n><data>   Define custom character.
   <n>         Character 0-7.
   <data>      8 bytes hex data.

   %zd1_0000040E04000000   Define custom character 1 as a 'bullet'.

   The normal size of a LCD character is 5*7pixels. AmigaLoad handles 8*8pixel
   characters, therefore only the upperleft 5*7pixels are used on normal
   LCD displays.

   Table to convert one line of a character to hex:

   Char     Hex      Char     Hex

   00000    00       10000    10
   00001    01       10001    11
   00010    02       10010    12
   00011    03       10011    13
```

```
00100    04       10100    14
00101    05       10101    15
00110    06       10110    16
00111    07       10111    17

01000    08       11000    18
01001    09       11001    19
01010    0A       11010    1A
01011    0B       11011    1B

01100    0C       11100    1C
01101    0D       11101    1D
01110    0E       11110    1E
01111    0F       11111    1F
```

The perfect place for custom character definitions is in the Init string. Avoid custom character definitions in the Main string, because the %zd command takes quite a long time to execute.

Example:

Character 'a':

| Line | Data | Hex |
|---|---|---|
| 0 | 00000 | 00 |
| 1 | 00000 | 00 |
| 2 | 01110 | 0E |
| 3 | 00001 | 01 |
| 4 | 01111 | 0F |
| 5 | 10001 | 11 |
| 6 | 01111 | 0F |
| 7 | 00000 | 00 (Not displayed for 5*7 caharcters) |

```
%zd1_00000E010F110F00
```

Now %z1 will print a 'a' on the LCD.

Character 'Bullet':

| Line | Data | Hex |
|---|---|---|
| 0 | 00000 | 00 |
| 1 | 00000 | 00 |
| 2 | 00100 | 04 |
| 3 | 01110 | 0E |
| 4 | 00100 | 04 |
| 5 | 00000 | 00 |
| 6 | 00000 | 00 |
| 7 | 00000 | 00 (Not displayed for 5*7 caharcters) |

```
%zd1_0000040E04000000
```

Now %z1 will print a 'Bullet on the LCD.

Note: The custom characters have ASCII code 0 to 7!
%z0 is defined as '\' and are used by %a!

%z<n> Write character with ASCII code <n> to LCD.
<n> ASCII code in hex. (00-FF).

```
   %z41  Writes 'A' to LCD.
   %z1   Writes custom character 1 to LCD. See %zd for more info!

   Note: The custom characters have ASCII code 0 to 7!
         %z0 is defined as '\' and are used by %a!
```

## 1.24 Update

```
%!       Update LCD.

   %!    Copys internal buffer to LCD.

   See also %d!

%d<n>    Update LCD and delay.
   <n>   (0-99) Delay <n>/10s. Default 1s.

   %d    Update LCD and delay 1s
   %d1   Update LCD and delay 0.1s
   %d0   Update LCD.

   See also %!!
```

## 1.25 Type

```
%s<n>    Title.
   <n>   Type number (1-4).

   %l0   Writes 'CPULoad' to LCD.
   %l1   Writes 'Ready' to LCD.

   See also %l, %r and %v!

%l<n><l> Left aligned title.
   <n>   Type number (1-4).
   <l>   String length. Default 8.

   %l0   Writes 'CPULoad ' to LCD.
   %l19  Writes 'Ready     ' to LCD.

   See also %s, %r and %v!

%r<n><l> Right aligned title.
   <n>   Type number (1-4).
   <l>   String length. Default 8.

   %l0   Writes ' CPULoad' to LCD.
   %l19  Writes '     Ready' to LCD.

   See also %s, %l and %v!
```

```
%v<n><l> Value (0-100%) for type <n>.
   <n>   Type number (1-4).
   <l>   Length. Default 3.

   %v1   Writes ' 20' to LCD.
   %v210 Writes '        80' to LCD.
```

## 1.26 Time/Uptime

```
%t       Time.

   %t    Writes '21:40' to LCD.

%u<n>    Uptime.
   <n>   Number of characters for 'days'. Default 3.

   %u    Writes '  1-07:42' to LCD.
   %u1   Writes '1-07:42' to LCD.
```

## 1.27 Text

```
%m<l>    Center string.
   <l>   Length of string.

   %m3Abc   Writes '      Abc       ' to LCD.
```

## 1.28 Gauge

```
%g<n><l/r><l>  Gauge.
   <n>   (1-4)     Type
   <l/r> (l/r/L/R) Align (Left/Right)
   <l>   (1-...)   Length

   %g1l10   Writes a left aligned gauge that is 10 characters long
            displaying the value of type 1. Userdefined characters
            are lost. See %z!
   %g2r8    Writes a right aligned gauge that is 8 characters long
            displaying the value of type 2. Userdefined characters
            are lost. See %z!
   %g1L10   Writes a left aligned gauge that is 10 characters long
            displaying the value fo type 1. Userdefined characters
            are used! See %z!

   Custom gauge (Advanced):
      When g<n>l and g<n>r are used the custom charcaters 1-7 are replaced
      with builtin character definitions. When g<n>L and g<n>R are used
      the custom characters are not changed.

      See also %zd!
```

```
Example:
   Custom left aligned gauge:

      Char1 Hex Char2 Hex Char3 Hex Char4 Hex Char5 Hex Char6 Hex Char7  ↩
         Hex
      00000 00  00000 00  00000 00  00000 00  00000 00  00000 00  00000  ↩
         00
      00000 00  10000 10  01000 08  00100 04  00010 02  00001 01  00000  ↩
         00
      00000 00  10000 10  01000 08  00100 04  00010 02  00001 01  00000  ↩
         00
      00000 00  10000 10  11000 18  11100 1C  11110 1E  11111 1F  11111 1 ↩
         F
      00000 00  10000 10  01000 08  00100 04  00010 02  00001 01  00000  ↩
         00
      00000 00  10000 10  01000 08  00100 04  00010 02  00001 01  00000  ↩
         00
      00000 00  00000 00  00000 00  00000 00  00000 00  00000 00  00000  ↩
         00
      00000 00  00000 00  00000 00  00000 00  00000 00  00000 00  00000  ↩
         00

      %zd1_0000000000000000
      %zd2_0010101010100000
      %zd3_0008081808080000
      %zd4_0004041C04040000
      %zd5_0002021E02020000
      %zd6_0001011F01010000
      %zd7_0000001F00000000

      %g1L10   Writes a left aligned gauge that is 10 characters long.

   Custom right aligned gauge:

      Char1 Hex Char2 Hex Char3 Hex Char4 Hex Char5 Hex Char6 Hex Char7  ↩
         Hex
      11111 1F  11111 1F  11111 1F  11111 1F  11111 1F  11111 1F  11111 1 ↩
         F
      00000 00  00001 01  00011 03  00111 07  01111 0F  11111 1F  11111 1 ↩
         F
      00000 00  00001 01  00011 03  00111 07  01111 0F  11111 1F  11111 1 ↩
         F
      00000 00  00001 01  00011 03  00111 07  01111 0F  11111 1F  11111 1 ↩
         F
      00000 00  00001 01  00011 03  00111 07  01111 0F  11111 1F  11111 1 ↩
         F
      00000 00  00001 01  00011 03  00111 07  01111 0F  11111 1F  11111 1 ↩
         F
      11111 1F  11111 1F  11111 1F  11111 1F  11111 1F  11111 1F  11111 1 ↩
         F
      00000 00  00000 00  00000 00  00000 00  00000 00  00000 00  00000  ↩
         00

      %zd1_1F00000000001F00
      %zd2_1F01010101011F00
      %zd3_1F03030303031F00
```

```
            %zd4_1F07070707071F00
            %zd5_1F0F0F0F0F0F1F00
            %zd6_1F1F1F1F1F1F1F00
            %zd7_1F1F1F1F1F1F1F00

            %g1R10   Writes a right aligned gauge that is 10 characters long.

         Note: The custom character definition is too long to fit in the
               Init string. Atleat one of the definitions (%zd1) must be
               moved to the Start string!

               When you have custom character definitions in the Start
               string you must press 'Test Start' to update your characters!

   See also %v!
```

## 1.29 Clear

```
%c       Clear LCD + Home

   %c    Clear LCD and move cursor to (0, 0).

   Note: The LCD-display is not updated. Use %! or %d!
```

## 1.30 Special

```
%a<n>    Writes a rotating 'bar'.
   <n>   (None/r) Rotation direction.

   %a    | / - \ | / - ...
   %ar   | \ - / | \ - ...

%o<l/r/u/d> Scroll
   %ol   Scroll LCD 1 character left.
   %ou   Scroll LCD 1 row up.
   %or   Scroll LCD 1 character right.
   %od   Scroll LCD 1 row down.

   Note: LCD is not updated! See %! and %d!

%*       Comment. Removes all text between %* and %*.
         If odd numer of %* text is removed until end of line.

   Abc%*cde%*fgh  Writes 'Abcfgh' to LCD.

%%       Write '%' to LCD.

   %%    Writes '%' to LCD.
```

## 1.31 LCD Examples

```
Example 1:
   Uptime and numeric. (Default)

   Start:
      %m7Welcome%d_%c%m2to%d_%c%m14AmigaLoad 2.0!%d

   Init:

   Main:
      Uptime : %u1_%h%y1%l17:    %v1%%

   End:
      Have a nice day!%d

Example 2:
   Uptime and left aligned gauge.

   Type 1: New CPULoad

   Start:

   Init:

   Main:
      Uptime : %u1_%h%y1CPU:[%g1l10]

   End:

Example 3:
   Uptime and custom right aligned gauge.

   Type 1: New CPULoad

   Start:
      %zd1_1F00000000001F00

   Init:
      %zd2_1F01010101011F00%zd3_1F03030303031F00%zd4_1F07070707071F00
      %zd5_1F0F0F0F0F0F1F00%zd6_1F1F1F1F1F1F1F00%zd7_1F1F1F1F1F1F1F00

   Main:
      Uptime : %u1_%h%y1CPU:[%g1R10]

   End:
```

## 1.32 String

```
AmigaLoad LCD Strings are written in this order:

AmigaLoad starts

   Clear LCD
   Write Start string to LCD
   Update LCD
```

```
   {
      Clear LCD
      Write Init string to LCD

      Loop
      {
         Write Main string to LCD
         Update LCD
         Delay
      }
   }
   Clear LCD
   Write End string to LCD
   Update LCD

AmigaLoad quits
```

## 1.33 MCI/LED

```
Short explanation of the gadgets...

Instr:      Set to 'MCI' if you have a MCI conneted and 'LED (10)' if you
            have a 10 LED-display connected.

Type:       What should the MCI/LED show? Short description.

Freq:       The frequency of the signal that is send to the MCI/LED.
            This frequency should be as low as possible. Decrease
            the frequency until the pointer on the MCI almost starts
            to vibrate (~10Hz). Default = 20Hz.
            Set to 10-20Hz for LED-displays.

Int Pri:    The priority of the timer interrupt that generates the
            square wave for the MCI/LED. If set too high it may disturbe
            other 'more important' interrupts. Default = -16.

Calibrate:  Calibrate the MCI/LED. See Calibrate MCI and/or Calibrate LED!

Test Cal:   Test the calibration. The slider value should be the
            same as the MCI/LED value.

Test NoCal: Test the MCI/LED. The slider value will not be the same
            as the MCI/LED value. When the slider is 100% the output
            is ~4V DC.
```

## 1.34 ARexx

```
AmigaLoad 2.0 understands some ARexx commands.

The port name is: AMIGALOAD

Commands (See also MUIs ARexx doc):
```

```
 GetTitle   : Get instrument title
 GetValue   : Get instrument value
 SetValue   : Set instrument value
 GetUpdate  : Get update rate

For examples see scripts in AmigaLoad/ARexxScr drawer!"
Example 1: ARexxScr/List.rexx.
Example 2: ARexxScr/SetValue.rexx.
```

## 1.35 GetTitle

```
Get instrument title.

GetTitle <type>

<type>
   NEWCPULOAD
   CPULOAD
   READY
   PUBLICMEM
   FASTMEM
   CHIPMEM
   FRAGFAST
   FRAGCHIP
   DEVICE1
   DEVICE2
   DEVICE3
   STACK1
   STACK2
   STACK3
   AREXX1
   AREXX2
   AREXX3

Returns instrument title in 'result'.
```

## 1.36 GetValue

```
Get instrument value.

GetValue <type>

<type>
   NEWCPULOAD
   CPULOAD
   READY
   PUBLICMEM
   FASTMEM
   CHIPMEM
   FRAGFAST
   FRAGCHIP
```

```
   DEVICE1
   DEVICE2
   DEVICE3
   STACK1
   STACK2
   STACK3
   AREXX1
   AREXX2
   AREXX3

Returns instrument value (0-100%).

Note: CPULOAD will return garbage values if CPULOAD isn't used
      somewere else (Virtual, MCI, LED or LCD).
      Use 'New CPULoad' instead!
```

## 1.37 SetValue

```
Set instrument to <value>.

SetValue <type> <value>

<type>
   AREXX1
   AREXX2
   AREXX3

<value>
   0-100
```

## 1.38 GetUpdate

```
Get update rate.

GetUpdate

Returns the value in Common/Update * 10 in 'result'.
```

## 1.39 Calibrate MCI

```
If you have one or more MCI:s connected:

Note: 'MCI 1' settings are for pin 9, and 'MCI 2' for pin 5!

Select 'MCI 1' or 'MCI 2'.

Set the cyclegadget 'Instr' to 'MCI'.

The first thing you should do is to click on the 'Test NoCal' gadget.
The pointer on the MCI should move when you change the value of the
```

slider. If the pointer doesn't move - turn off your computer and check
the hardware. If the MCI never reach 100% or the MCI reach 100% when
the slider is <85% read the Hardware MCI doc.

The MCI must be calibrated before you can use it:

   Click on the 'Calibrate' gadget.

   Make sure that the slider is set to 0% and use the screw on the
   MCI to adjust the pointer to 0%. Increase the slider value until
   the pointer on the MCI starts to move. Click on the 'Ok' gadget.

   Use the slider (the slider value is unimportant) to set the pointer
   on the MCI to 10%. Click on the 'Ok' gadget.

   And so on...

   Test the calibration with 'Tast Cal' (Slider value and MCI pointer
   should display the same value).

## 1.40 Calibrate LED

If you have one or more LEDs connected:

Note: 'MCI 1' settings are for pin 9, and 'MCI 2' for pin 5!

Select 'MCI 1' or 'MCI 2'.

Set the cyclegadget 'Instr' to 'LED (10)'.
If you have more/less than 10 LEDs use the 'MCI' mode instead. In 'LED (10)'
mode the calibration data will be used directly without any intepolation,
this prevents the LEDs from flickering when the output 5%, 15%, 25%,...

The first thing you should do is to click on the 'Test NoCal' gadget.
Now it should be possible to control the LEDs with the slider,
if you can't - turn off your computer and check the hardware.
If the last 2-3 LEDs doesn't work as expected or if the LED-display
reach 100% when slider < 50% - read the Hardware LED doc.

The LED must be calibrated before you can use it:

   Click on the 'Calibrate' gadget.

   Use the slider to find the value midway 0% and the slidervalue when
   the first LED is almost active. Click on the 'Ok' gadget.

   Use the slider to find the value midway:
      Value when the first LED is almost active.
   and
      Value when the second LED is almost active.
   Click on the 'Ok' gadget.

   Use the slider to find the value midway:
      Value when the second LED is almost active.
   and

```
   Value when the third LED is almost active.
Click on the 'Ok' gadget.

And so on...

Test the calibration with 'Tast Cal'. Slider value and the LEDs
should display the same value, and none of the leds should 'flicker'.
```

## 1.41 Type

```
NONE:          Always 0%.

New CPULoad:   Processor load.
                 0% CPU has nothing to do.
               100% CPU always busy.

CPULoad:       How much time does a -128 pri task get?
                 0% CPU has nothing to do.
               100% CPU always busy.

               Obsolete! Use 'New CPULoad'!

               Note: You can't use 'CPULoad' and 'New CPULoad at the same time ↩
                  '.

Ready:       How much CPU time does a new 0 pri task get?
                 0% A new task will get almost no CPU.
               100% A new task may use 100% of the CPU.

Public Mem:    How much public memory is free?
                 0% Out of public memory.
               100% All public memory free.

Fast Mem:      How much fast memory is free?
                 0% Out of fast memory.
               100% All fast memory free.

Chip Mem:      How much chip memory?
                 0% Out of chip memory.
               100% All chip memory free.

Frag Fast:     How large is the biggest fast-memory block
               compared with total free fast-memory?
                 0% No fragmentation
               100% Your fast-memory is fragmentated...

Frag Chip:     How large is the biggest chip-memory block
               compared with total free chip-memory?
                 0% No fragmentation
               100% Your chip-memory is fragmentated...

Device 1-3:    How much space is used on the device?
                 0% Device is emty or not found
               100% Device is full
```

```
                Note: You will get a requester if no disk was found...

Stack 1-3:      How much space is used of the stack?
                  0% Stack is emty or task not found
                100% Stack is full

                Warning: AmigaLoad does nothing to prevent stack overflow!
                         If stack overflows AmigaLoad will still display 100%.

ARexx 1-3:      Controlled by ARexx command SetValue AREXX1-3.

Note: If 'Invert' is selected all values are inverted.
```

## 1.42 BC547B

```
TO92              Not important
Type = NPN        Important
Ptot = 0.3W       >0.05W
Vceo = 45         >5
Ic   = 0.1A       >0.01A
ß    = 200/450    Not important
ft   = 300        Not important


 _____
/_____\
|     |
|BC547|
-------
|  |  |
|  |  |
|  |  |
|  |  |
1  2  3

1 = Collector
2 = Base
3 = Emitter
```

## 1.43 Bugs

When Uptime and/or Time 'Window Title' is selected, the windows title gadgets may be overwritten.

24byte memory loss when MCI #1 is used.

If you find any bugs, please e-mail me (tl94hfl@student.hgs.se)!

## 1.44 Hardware MCI

What you need:

```
   1 Moving coil instrument (Ammeter)
   1 9-pin D-Sub Female
   1 Transistor BC547B (or equvivalent) T1
   1 Capacitor ~100$\mathrm{\mu}$F              C1
   1 Resistor  ~10kohm              R1
   1 Resistor  ~50kohm - ~10Mohm    R2
   1 Resistor  ~100ohm - ~7kohm     R3
   1 Resistor  ~100ohm - ~1kohm     R4
```

The MCI should have a scale that is easy converted to percent. Ex:

```
   0 - 100$\mathrm{\mu}$A
   0 -   1mA
   0 -  10mA
   0 - 100mA
   0 -    1A
```

Moving Coil Instrument = MCI

The output from pin 5/9 on the joystickport is a square wave (0 - 4V) with a duty-cycle from 0% to 100% when AmigaLoad is running.
(Use 'Test NoCal' in AmigaLoadSettings to test)

'MCI 1' settings are for pin 9, and 'MCI 2' for pin 5!

Construction help: For more info see picture AmigaLoadMCI.iff!

Imax = Maximum current for MCI. See table!
Ic = Collector current.
Ib = Base current.
Ri = Internal resistanse for the MCI.
ß = Ic / Ib. For BC547B ß = 200.

| Min | Max | Imax | Ri | R2 | R3 | R4 |
|---|---|---|---|---|---|---|
| 0 - 100$\mathrm{\mu}$A | 100$\mathrm{\mu}$A | ~800ohm - ~4000ohm | | 1M | 120 | 140 - 702 |
| 0 - 1mA | 1mA | ~70ohm - ~180ohm | | 670k | 120 | Remove |
| 0 - 10mA | 10mA | ~3.5ohm | | 67k | 120 | Remove |
| 0 - 100mA* | ~7mA | ~7ohm | | 96k | 120 | Remove |
| 0 - 1A* | ~7mA | ~7ohm | | 96k | 120 | Remove |

*) Remove internal/external shunt!

The LP-filter converts the square wave to DC-voltage. I have used a simple first order filter, but you can use a more advanced if you like.
I have used Tau = R1*C1 = 10k*100$\mathrm{\mu}$ = 1s.

R2 must be bigger then R1, else the LP-filter will not work as expected.
R2min = 10*R1 = 10*10k = 100kohm

R3 is not necessary in all situations. It's main purpose is to avoid short-circuit if you connect anything wrong. R3 is the only component connected to +5V if you have connected everything correct.

R4 = R2*Ri*Imax / ( (4 - 0.65)*ß - R2*Imax )  Equ 1

```
Try R2 = 1M in equ 1

If R4 > 100ohm
{
   Use   R1 =  10kohm
         R2 =   1Mohm
         R3 = 120ohm
         R4 = Calculated value from Equ 1
         C1 = 100$\mathrm{\mu}$F
}
else
{
   R2 = (4 - 0.65)*ß / Imax                    Equ 2
   If R2 < 100kohm, test with another value in Equ 1.
   Use   R1 =  10kohm
         R2 = Calculated value from Equ 2
         R3 = 120ohm
         R4 = Infinite = Remove
         C1 = 100$\mathrm{\mu}$F
}

You will probably not find resistors with the calculated values.
Available resistors are:
    10,  12,  15,  18,  22,  27,  33,  39,  47,  56,  68,  82,
   100, 120, 150, 180, 220, 270, 330, 390, 470, 560, 680, 820,
   ...
   ...
   ...

Always choose the resistors so the current gets larger than the
calculated value ( R2 and R3 smaller and R4 bigger ).

+5V  should be connected to pin 7,
Gnd  should be connected to pin 8 and
Data should be connected to pin 9 (MCI 1) or pin 5 (MCI 2) on the D-Sub.

Build the circuit and put it in a nice box.

Connect the D-sub to the joystickport. Have the power off when you
insert the D-sub. When you start your computer, make sure that the
power led shines and that it doesn't come smoke from your computer,
if it does - something is very wrong...

Start AmigaLoad and AmigaLoadSettings.
Click on the 'Test NoCal' gadget in AmigaLoadSettings.
The pointer on the MCI should move when you change the value of the
slider. If the pointer doesn't move - turn off your computer and check
the hardware. If the MCI never reach 100% or the MCI reach 100% when
the slider is <85% change the values on R2, R3 and/or R4.

When AmigaLoad is not running the MCI may show ~20% anyway, this
is normal and nothing to worry about...

If you have problems building the hardware or to calculate the resistor
values - write a mail to me, and I will try to help you!
```

## 1.45 Hardware LED

```
What you need:

   1 IC LM3914 (Dot/Bar Display Driver) or similar
   1 9-pin D-Sub Female
  10 LEDs
   1 Capacitor ~47$\mathrm{\mu}$F                      C1
   1 Resistor  ~10kohm                  R1
   1 Resistor    10kohm, 1.8kohm or 1kohm R2
   1 Resistor   8.2kohm, 1.8kohm or 1kohm R3

Light Emitting Diod = LED

Thanks to Yannick Erb for the LED hardware!

The output from pin 5/9 on the joystickport is a square wave (0 - 4V) with
a duty-cycle from 0% to 100% when AmigaLoad is running.
(Use 'Test NoCal' in AmigaLoadSettings to test)

'MCI 1' settings are for pin 9, and 'MCI 2' for pin 5!

Construction help: For more info see picture AmigaLoadLED.iff!

The LP-filter converts the square wave to DC-voltage. I have used a
simple first order filter, but you can use a more advanced if you like.
I have used Tau = R1*C1 = 10k*100$\mathrm{\mu}$ = 1s.

LM3914 can work in two different modes - Dot and Bar.
In Dot-mode only one LED is active at the same time.
The mode is controlled with pin 9 on the LM3914. Connect to +5V for
Bar-mode, and leave unconnected for Dot-mode (Not GND).

The LED brightnes is controled with the two resistors R2 and R3:
                                  Bar-mode:    Dot-mode:
   R2 =  10kohm and R3 = 8.2kohm -> Itot =  50mA Itot =  5mA ILED =  5mA
   R2 = 1.8kohm and R3 = 1.8kohm -> Itot = 100mA Itot = 10mA ILED = 10mA
   R2 = 1.0kohm and R3 = 1.0kohm -> Itot = 160mA Itot = 16mA ILED = 16mA

Note: Maximum current (Itot) for some joystickports are 125mA (A500)!

R2 and R3 also defines the voltage when all 10 LED:s are on. This
voltage must be smaller than 5-1.8 = 3.2V or the last 2-3 LEDs won't
work as expected. Use a smaller R3 if you have this problem.

+5V  should be connected to pin 7 on the D-Sub,
Gnd  should be connected to pin 8 on the D-Sub and
Data should be connected to pin 9 (MCI 1) or pin 5 (MCI 2) on the D-Sub.

Build the circuit and put it in a nice box.

Connect the D-sub to the joystickport. Have the power off when you
insert the D-sub. When you start your computer, make sure that the
power led shines and that it doesn't come smoke from your computer,
if it does - something is very wrong...
```

```
Start AmigaLoad and AmigaLoadSettings.
Click on the 'Test NoCal' gadget in AmigaLoadSettings.
Now it should be possible to control the LEDs with the slider,
if you can't - turn off your computer and check the hardware.
If the last 2-3 LEDs doesn't work as expected   - use smaller R3.
If the LED-display reach 100% when slider < 50% - use larger R3.

When AmigaLoad is not running the LEDs may show ~20% anyway, this
is normal and nothing to worry about...

If you have problems building the hardware or to calculate the resistor
values - write a mail to me, and I will try to help you!
```

## 1.46 Hardware LCD

```
Liquid Crystal Display = LCD

To use a LCD-display with AmigaLoad you need LCDaemon by Hendrik De Vloed.
You can download LCDaemon from Aminet.

All LCD-display hardware information can be found in the LCDaemon documentation ↩
    !

Send all questions/suggestions about the LCD-display HARDWARE to
Hendrik De Vloed (hendrik.devloed@barco.com).

Note: You can try different LCD-display sizes with lcd_ami from LCDaemon
      before you buy/build the hardware!
```