


# Hacklace Software

Your Hacklace comes pre-programmed with the Arduino bootloader and the Hacklace firmware, so you can use it right after assembly. The following paragraphs show you how to modify the Hacklace according to your needs.

[There is a separate page for programming the Hacklace.](#)

## Downloading Texts and Animations



 Hacklace2 with Hack-Key

A textfile is used to transfer your own ticker texts and pixel art animations to the EEPROM of the Hacklace. The firmware in the flash memory remains unchanged. Connect the Hacklace to a PC using a USB-to-serial-converter (see ["Connecting to a PC"](#)). You may start by modifying the [default configuration file](#) and downloading it to the Hacklace.

There is a short video that illustrates the first steps.



## Video

For conversion of standard image file formats find a tool in the [download section](#).

## Download-Mode

Before the Hacklace can accept data it has to be put into download mode. This can be achieved in two ways:

- Trigger a reset. If you use a [Hack-Key](#) this will happen automatically if you open the serial connection.
- During power-on hold the right push button pressed.

When the Hacklace is in download mode it displays „DL“. Now you can send characters to the Hacklace. To provide some feedback every received (printable) character will be shown on the display. If all the space in the EEPROM is used up further reception of characters will result in the display of „EE“ to indicate a memory overflow error. You can leave download mode at any time by pressing the left button. The Hacklace will then switch back to normal operation and display the first text, animation or app.

## Programs for the serial interface

There are many programs for sending data to the hacklace using the serial interface. For a start you can try the „serial monitor“ that is integrated into the Arduino-IDE. Make sure to select the correct baud rate (2400). Now you can send simple command strings.

In order to download complete configuration files to the Hacklace it is more convenient to use one of the following programs.

- Hyperterminal (Windows XP)
- Coolterm (Windows, Linux, OS X)
- HTerm (Windows 7 / 8)

If you are working under a UNIX-like operating system you might prefer the command line. Set up the correct transfer parameters and send strings using the printf command.

```
stty -f /dev/ttyUSB0 2400
printf '\eHL\\00 85 01\\ Hello World!  \\00 FF\\' > /dev/ttyUSB0
```

## Commands

The escape character <ESC> (character code 27) will always cause the receiver of the Hacklace to reset. It is good practice to start a data transfer by sending <ESC>. After that a command of two capital letters will follow.

- „HL“ (Hacklace) - Transfer text and animation data to the EEPROM.
- „RD“ (restore defaults) - Fill the EEPROM with the default content.
- „VN“ (version number) - Show the version number of the Hacklace firmware.

## Data Format

The EEPROM keeps a list of app entries. In normal mode the Hacklace will follow this list when browsing the apps by pressing the left button. The first byte of each entry contains the app-ID, which is a number that identifies the app that is to be called. The next bytes can be parameters that are read in and evaluated by the app. The number of parameter bytes depends on the app.

**Important:** The app-list must be terminated by app-ID 255 (FF hexadecimal). Forgetting this will

cause „bad ID“ errors.

### Storing characters / non-printable characters

After reception of the „HL“ command all following printable characters (character code  $\geq 32$ ) will be stored in EEPROM. Non-printable characters will be ignored. This means that you can format your textfile using non-printable characters like tabs and empty lines.

### Entering hex values

Data can also be entered as hexadecimal values. For this they have to be presented as two-digit hexadecimal numbers enclosed by '\'. Several bytes have to be separated by a space (to be precise: by a non-hexadecimal character). With the hex input feature you are able to store codes below 32 to access special characters or to enter data for animations. If you need the backslash as an ordinary character you have to double it. Examples:

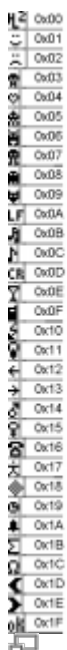
- The character string 'JKLM' can also be entered as '\4a 4b 4C 4D\'.
- With '\05 06 07 08\' you access the special characters chr(5) to chr(8). They will show up as space invader monsters.
- The sequence '\\ produces the character '\' = chr(92).

### Comments

The sequence '\#' causes the Hacklace to ignore the rest of the line. It is used for adding single-line comments to your textfile.

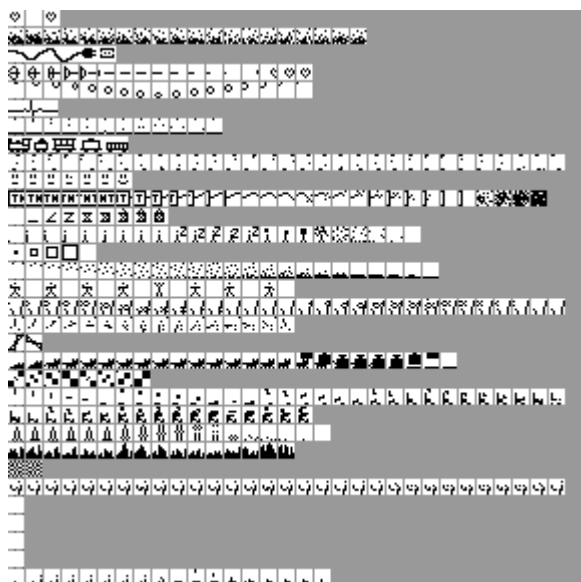
**Important:** Use <TAB> instead of <SPACE> characters to separate comments from the rest of the line. Spaces are printable characters and will be stored in EEPROM, which causes „bad ID“ errors.

## Character Font



The Hacklace has an international character font according to ISO8859-15 which is well suited for displaying Western-European languages. It is a proportional font, i. e. the character width varies. In addition it has a fancy set of special characters.

- character 0: system control character (do not use)
- character 1-30: special characters
- character 31: system control character (do not use)
- character 32 - 127: standard ASCII characters
- character 128 - 159: data for preset animations
- character 160 - 255: national characters according to ISO8859



fertige Animationen

## Apps

The Hacklace may have up to 32 apps. Each app is uniquely identified by its app-ID. The app-ID is defined by its position in HL\_AppRegistry.h. The Hacklace software already contains several apps that are ready to use and can serve as examples for programming your own apps.

### Animation-App

| app-ID | speed + delay | dir + inc | text / animation data | end flag |
|--------|---------------|-----------|-----------------------|----------|
| 0      | 1 byte        | 1 byte    | n bytes               | 0        |

This is probably the most used app of all. It is the one that displays ticker texts and animations. When this app is executing the right push button can be used to increase the display brightness in four steps.

### Parameters

- The upper **nibble** of the speed+delay parameter contains the scrolling speed. The lower nibble defines the duration of the pause at the end of a scrolling cycle. Pause duration is proportional to the scrolling speed.
- The upper nibble of the dir+inc-parameter defines the scrolling direction. Possible values are 0 (forward), 1 (backward) and 2 (bidirectional). The lower nibble specifies the step width for

scrolling. Meaningful values are 1 for ticker texts and 8 for animations.

- The data part can be of variable size. It has to be terminated by 0 (end flag). Every byte will be interpreted as a character code which is displayed according to the character font. For entering data directly without using the font (e. g. for animations) you have to use a binary data block. The block starts with the hex value 1F. Then comes the number of data bytes that will follow (up to 255) and finally the data itself. Each byte represents a display column. The eight bits correspond to the eight pixels of the column. Bit0 controls the state of the top pixel, bit7 the bottom pixel.

Thanks to SubOptimal there is a [command line tool](#) available that converts common image formats to Hacklace2 hex byte strings. You can use this tool directly or its sourcecode may serve as an example for your own developments.

Johannes has developed an [Excel- / Calc-sheet](#), with which you can easily design pixel graphics and generate the corresponding hex codes.

### Example

```
\00 7A 01\It's hip to be \1F 08 00 7E 42 42 42 42 7E 00\! \00 FF\
```

This string is interpreted as follows:

|                         |   |
|-------------------------|---|
| 00                      | --> app-ID (animation app)                          |
| 7A                      | --> speed=7, delay=10 (= A hex)                     |
| 01                      | --> scrolling forwards (0), one column per step (1) |
| "It's hip to be "       | --> text input                                      |
| 1F                      | --> start of binary data block                      |
| 08                      | --> number of data bytes (8)                        |
| 00 7E 42 42 42 42 7E 00 | --> data ("a square")                               |
| "! "                    | --> text input                                      |
| 00                      | --> end flag  |
| FF                      | --> app-ID 255 = end of app-list                    |

### Spiral-App

| app-ID |
|--------|
| 1      |

This app produces a spiral pattern. It has no parameters.

### Counter-App

| app-ID |
|--------|
| 2      |

A simple hand counter. By pressing the right button the counter increases by one. Holding the right button pressed will reset the counter. This app has no parameters.

## Game-of-Life-App

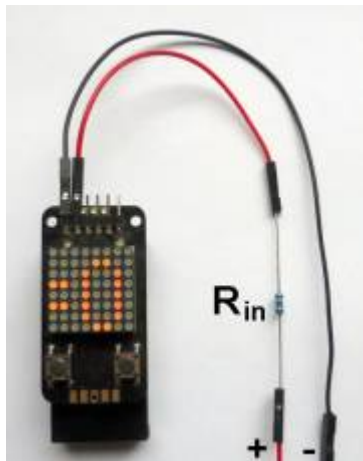
| app-ID | iterations | interval | start configuration |
|--------|------------|----------|---------------------|
| 3      | 1 byte     | 1 byte   | 8 bytes             |

An implementation of Conway's [W](#)"Game of Life".

### Parameters

- The first parameter gives the number of simulation cycles. After this number of cycles the game re-starts with the start configuration.
- With the interval parameter you can influence the simulation speed by specifying the time interval between two simulation cycles. It will be  $\text{interval} \times 10$  milliseconds.
- The start configuration is defined by 8 bytes. Each byte corresponds to a display column, the bits of which define the cell state (1 = living, 0 = dead).

## Voltmeter-App



Hacklace measuring voltages

| app-ID | input resistance |
|--------|------------------|
| 4      | 1 byte           |

With this app you can measure voltages. Connect the reference potential to pin 1 of the communication port (GND). Connect the unknown voltage with a series resistor to pin 2. Its resistance is specified in the parameter. Together with R14 (= 10 k $\Omega$ , see [schematic](#)) the resistor forms a voltage divider which defines the voltage range that can be measured. Without a resistor (resistance = 0  $\Omega$ ) the input range of the A/D-converter is 0 - 1.1 volts. With a series resistor of 39 k $\Omega$  the input range is 0 - 5 volts. To protect the input it is recommended to put a zener diode of 3.6 volts between pin 1 and pin 2 (cathode at pin 2).

### Parameter

- resistance of input series resistor (in k $\Omega$ )

## Battery-Monitor-App



Add two resistors to monitor the battery voltage.

|               |
|---------------|
| <b>app-ID</b> |
|---------------|

|   |
|---|
| 5 |
|---|

With the battery monitor the Hacklace can check the voltage of its battery. Depending on the voltage level a battery symbol which is more or less filled is displayed. For the battery monitor to work you have to add two resistors to the sensor port. Most easily fit smd resistors of size 0805. Solder 22 k $\Omega$  between pin 1 and pin 2 and 10 k $\Omega$  between pin 4 and pin 5.

## Frequency-Meter App

|               |
|---------------|
| <b>app-ID</b> |
|---------------|

|   |
|---|
| 6 |
|---|

This app measures frequencies. The signal should be a square wave and is connected to pin 5 of the sensor port (PD3). The input range is from 16 Hz to approx. 10 kHz. The app has no parameters.

## Thermometer-App

|               |
|---------------|
| <b>app-ID</b> |
|---------------|

|   |
|---|
| 7 |
|---|

With only a single resistor and a NTC thermistor the Hacklace is transformed into a thermometer. Temperature is displayed with two digits. The app has no parameters.

## Stopwatch-App

|               |
|---------------|
| <b>app-ID</b> |
|---------------|

|   |
|---|
| 8 |
|---|

The right pushbutton starts and stops the stopwatch. The display shows seconds and tenths of a second as a bargraph. A longer press of the right button resets the stopwatch. The app has no parameters.

## Speedometer-App

|               |                      |
|---------------|----------------------|
| <b>app-ID</b> | <b>circumference</b> |
|---------------|----------------------|

|   |
|---|
| 9 |
|---|

|         |
|---------|
| 2 bytes |
|---------|

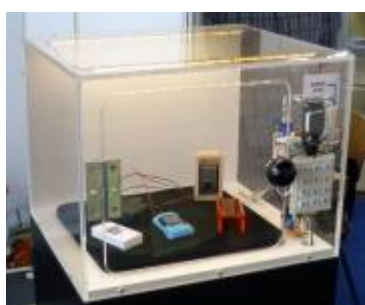
Determines the frequency at pin 5 of the sensor port and uses it to calculate the speed. For this the circumference of the wheel (in millimeters) must be given as a parameter. The parameter is 16-bit (low-byte, high-byte).

## Jukebox-App

| app-ID |
|--------|
| 12     |

This app plays little tunes using the pwm functionality of pin 5 of the sensor port. The right button switches to the next melody. The app has no parameters.

## Codelock-App



Safe with a Hacklace codelock

| app-ID | number of valid codes | codes            |
|--------|-----------------------|------------------|
| 20     | 1 byte                | 2 bytes per code |

Attach a numeric keypad to the analog input of the sensor port. The digital output controls a standard servo motor. To open the lock enter a 4-digit code and press „#“. After three failed attempts you have to wait for some time. If the code is correct the servo unlocks the door. Pressing „\*“ locks the door again. Each valid code is stored as a 2-byte parameter where each [Nibble](#) represents one of the four digits. Default codes for opening are 1234 and 9999.

## Displaytest-App

| app-ID |
|--------|
| 21     |

All rows and columns of the display will turn on one by one. This happens repeatedly at minimum and maximum brightness setting. The app has no parameters.

## Example-App

| app-ID |
|--------|
| 30     |

The Example-App has no parameters. It generates some simple graphics (ASCII characters) and modulates the brightness of the display. Use this app as a starting point for programming your own apps.



## Reset-App

| app-ID |
|--------|
| 31     |

The Reset-App is not actually an app but a placeholder. The app that is entered at the position of the Reset-App (app-ID 31) will be executed after a system reset. Usually this is the Download-App. For example when you use the [Hack-Key](#) opening the serial connection will trigger a reset. This will automatically start the Download-App putting the Hacklace into download mode. The Reset-App must be parameter-less.

## Display-App

With the Display-App the Hacklace can be used as a serial display. Set the baudrate (9600 baud) and scrolling speed in the source code if you need to adapt it. All characters received by the Hacklace are displayed as a string. <CR> or <LF> clears the display. There is also a raw mode available. If you precede a byte with the hex value 1F it is written directly into the display memory and not interpreted as a character code. The Display-App has no fixed app-ID. Often you like to put it at the position of the Reset-App (app-ID 31) so that after a reset the Hacklace starts operating as a display.

## Programming

You'd like to go beyond ticker texts and pixel animations? Develop your own apps or learn how to modify the Hacklace firmware. For this you need the [Arduino development platform](#) and a USB-to-serial-converter like the [Hack-Key](#). Find all the details on the [Hacklace programming page](#).

[Hacklace](#), [electronics](#), [kit](#), [software](#), [app](#)

From:

<http://www.doku.fab4u.de/> - **fab4U**

Permanent link:

<http://www.doku.fab4u.de/en/kits/hacklace/software>

Last update: **2018/07/03 21:12**

