1   usburn for Linux

usburn is a Linux program, that analyses a HEX-file and transfers the contained data via USB to the PIC-programmer Brenner8 or Brenner9. Beside this usburn can be used to :

- calibrate the Brenner8
- load new firmware into Brenner8/9 (with Bootloader)

1.1   Requirements for the use of usburn

1.1.1   Software

US-Burn runs under Linux with libusb. I test the function of the software on Debian, but it should word under all Linux versions.

1.1.2   Data

US-Burn needs input data (HEX-file) in Intel-Hex8-Format.

1.1.3   Hardware

The software supports the USB-Port-programmer „Brenner8" and „Brenner9" .

- Brenner8
- Brenner8mini
- Brenner9

Brenner8:   standard programmer with ICSP-connector and testsocket
http://www.sprut.de/electronic/pic/projekte/brenner8/index.htm

Brenner8mini: A simplified Brenner8 with ICSP connector
http://www.sprut.de/electronic/pic/projekte/brenner8mini/index.htm

Brenner9:   A programmer for 3,3V-PICs with ICSP connector
http://www.sprut.de/electronic/pic/projekte/brenner9/index.htm

1.2   Installation

Usburn requires libusb (http://libusb.wiki.sourceforge.net/), this can be installed by the packet manager of the Linux distribution..

The source code of usburn is part of a TAR-Filet. This file is named usburnxx.TAR and contains:

- Source code files   the program itself
- readme.txt   short description (German)
- xxx03.dat - files   the PIC-Database files
- readme_db.txt   short description of Database (German)
- make   make-File

- b8_handbuch.pdf   Handbook (German)
- b9_handbuch.pdf   Handbook (German)
- calibration_under_linux.pdf   English calibration instruction
- firmware files

First one has to extract the TAR-file, ("`tar xfv [ARCHIVNAME].tar`") and to compile the source code ("**make**").

Usburn uses libusb instead of kernel modules to access the programmer, consequently **root** privileges are required. To give access to normal users one has to create a file "**/etc/udev/rules.d/99-sprutbrenner**" and to write into this file:

```
SUBSYSTEM=="usb", SYSFS{idProduct}=="ff0b", SYSFS{idVendor}=="04d8",
GROUP = "plugdev"
```

If the programmer is now connected to the PC, all members of group plugdev (should be all normal users) have access to the programmer.

1.3   Use usburn

Usburn is a command line program; it is controlled by command line options (with parameters).

Usburn understands long-options and short-options with and without parameters.

Every long-option is one word with a "**--**"-prefix (2 minus signs). If a parameter is required, then option and parameter are separated by one space.
("**--FAMILY 18**")

Short-options are one letter with a "**-**"-prefix (one minus sign). If a parameter is required, then if follows directly. No space is needed for separation. ("**-F18**")
Multiple short options (without parameter) can follow after a single "-".

Usburn is case sensitive.

1.4   Options

Options without Parameters

-h	--help	show the help-screen
-r	--read	read PIC-content into HEX-file
-w	--write	write HEX-file into PIC
-c	--compare	compare PIC-content with HEX-file
-e	--erase	erase PIC
-p	--remove	remove code protection from PIC
-i	--info	show a lot of unnecessary information

-d	--reanimate	reanimate a PIC, that don't reacts anymore
-l	--list	list all supported PIC-types
-a	--auto	autodetect PIC-type
-b	--blank	check if the PIC is blank
-o	--boot	switch programmer into bootloader-mode
-f	--firmware	update firmware
-n	--normal	deactivates bootloader-mode
-u	--run	activate Vdd for target-PIC
-k	--calibration	Calibrate Vpp-generation of Brenner8
-t	--test	interactive test of the hardware

Options with Parameter

-S	--SOCKET	PIC-Socket(Brenner8 only) or ICSP- connector
-F	--FAMILY	PIC-family (core architecture)
-H	--HEX	name of HEX-file
-I	--IN	name of input HEX-file (for write & compare)
-L	--OSCCAL	Osccal-value for oscillator fine-tuning
-O	--OUT	name of output HEX-file (for read)
-P	--PIC	manual selection of the PIC-type

-h   --help   show the help-screen
Shows a list of all possible options of usburn. Can be combined with other options.
Example:
```
      usburn --help
      usburn -h
```

-r   --read   read PIC-content into HEX-file
Reads the content of a PIC and writes it into a hex-file. The default file name is "HexOut.hex". With the options –HEX (-H) or --OUT (-O) the filename can be changed. If a file of same name still exists, then it will be overwritten without a warning.
If –read is used in combination with --write, --erase or --remove, then --read will be executed after the other options. Thus one would read out the erased or rewritten PIC.
Example:
```
      usburn --read  --SOCKET ICSP  -F18 --OUT test.hex
      usburn -r  -S28 -F16 -Otest.hex
```

-w   --write   write HEX-file into PIC
Writes the content of a hex-file into a PIC. The default file name is "HexIn.hex". With the options –HEX (-H) or --IN (-I) the filename can be changed.
Before the write process starts, the PIC will be erased, and if codeprotection is active, then codeprotection will be removed. After the end of the write

process, usburn checks that the PIC was programmed correctly.
There is no need to user the --erase, --remove or --compare options
Example:
```
usburn --write  --SOCKET ICSP  -F18 --IN test.hex
usburn -w  -S28 -F16 -Itest.hex
```


-c   --compare   compare PIC-content with HEX-file
The content of a PIC will be compared with a HEX-file, and differences will be
shown. The default file of the HEX-file name is "HexIn.hex". With the options
–HEX (-H) or --IN (-I) the filename can be changed.
By default only the number of errors will be indicated, but if **--info** -option is
selected too, then for every single error the memory-address, the hex-file-
value and the PIC-memory-value will be shown.
Example:
```
usburn --compare  --SOCKET ICSP  -F18 --IN test.hex
usburn -ic  -S28 -F16 -Itest.hex
```


-e   --erase   erase PIC
The PICs memory will be erased. For some PIC-types this will not work if
codeprotection is activated. In this case **--erase** should be combines with **--
remove**.
Example:
```
usburn --erase  --SOCKET ICSP  -F18
usburn -e  -S28 -F16
```


-p   --remove   remove codeprotection from PIC
Active codeprotection of the PIC will be disabled. At the same time the Flash-
program memory and the EEPROM-data memory will be erased.
Erase of Configuration and user-ID is not guarantied for all PIC types, but can
be realized by the combination of **--remove** and **--erase**.
Example:
```
usburn --erase  --remove --SOCKET ICSP  -F18
usburn -pe  -S28 -F16
```


-i   --info   show a lot of unnecessary information
By default usburn shows only important information at the console. But with
the option --info it becomes talkative and puts out a lot of additional
information that under normal conditions that is not necessary and may
confuse the user. But during malfunction analysis it may be helpful.
Example:
```
usburn --info --compare  --SOCKET ICSP  -F18 --IN test.hex
usburn -ic  -S28 -F16 -Itest.hex
```


-d   --reanimate   reanimate a PIC, that don't reacts anymore
Shit happens, and under strange conditions sometimes a PIC may be
programmed in a way, that makes it impossible to access this PIC by a
programmer. That is most the result of a faulty configuration.

In such a situation the **--reanimate** option may help to "reanimate" the PIC.

To use this option, it is necessary to use the additional option **--PIC**, because the automatic PIC-identification can not work if the PIC is deaf like a post.
Example:
```
usburn --reanimate --SOCKET ICSP  -F18  --PIC PIC18F2450
usburn -d  --SICSP  -F16  -PPIC16F876
```


-l   --list   list all supported PIC-types
This option lists all PICs that are supported by the actual combination of software, database and firmware.
Example:
```
usburn --list
usburn -l
```


-b   --blank   check if the PIC is blank
This option checks if Flash program memory and EEPROM data memory are blank. During normal use this option is not necessary.
Only if one likes to give used PICs to an other person he may use it, to prove that his very-very-very-secret data was correctly erased by the --erase option.
Example:
```
usburn --blank  --SOCKET ICSP  -F18
usburn --b -S28 -F16
```


-o   --boot   switch programmer into bootloader-mode
This option activates the Bootloader of the programmer. For normal use this option is not necessary.
Example:
```
usburn --boot
usburn -b
```


-f   --firmware   upload new firmware
This option activates the Bootloader and uploads a new firmware into the programmer. If this worked without any problems, then the programmer is automatically set back into normal mode.
The firmware is a HEX-file. Its name should be cited with the option --HEX or --IN. (Without this option the software tries to find a firmware-file with the unusual name HexIn.hex).
Example:
```
usburn --firmware  --IN fw_12.hex
usburn -f  -Ifw_12.hex
```


-n   --normal   deactivates bootloader-mode
This option switches a programmer from Bootloader-mode back into normal mode. If an operational firmware is contained in the programmer, then it will work normal. But if not, then the programmer becomes no operational and can only brought back to Bootloader mode by the Bootloader-jumper.

For normal use this option is not necessary, because after normal firmware-update the programmer is switched back to normal mode automatically.
Example:

```
usburn --normal
usburn -n
```


-u   --run   activate Vdd for target-PIC
(Only Brenner8P, Brenner8miniP, Brenner9. Only PIC that are connected via ICSP-connector)
If this option is selected, then Vdd of the target PIC will be switched on and MCLR connected to Vdd after all other options are worked off. The PIC will then start to run its program. This makes sense only if the PIC is located inside a test circuit and connected to the programmer via the ICSP-connector. Because all other options are worked off first, it's possible to combine --write and --run. The HEX-file will be programmed into the PIC and the PIC will then be started.

Then usburn waits for the "Enter/Return"-Key. If the key is pressed, then Vdd will be switched off and usburn quits.
Example:

```
usburn --write --run  --SOCKET ICSP  -F18 --IN test.hex
usburn -wu  -S28 -F16 -Itest.hex
```


-k   --calibration   starts the calibration of Brenner8
(Only Brenner8)
This option starts the calibration of the Vpp-generation. For details see the calibration instruction document.
Example:

```
usburn --calibration
usburn -c
```


-t   --test   starts an interactive test of the hardware
This option can be used to test all signals of the programmer.
Example:

```
usburn --test
usburn -t
```


-S   --SOCKET   PIC-Socket (Brenner8 only) or ICSP- connector
If a target PIC is in the testsocket of the programmer, then usburn needs to know the size of the PIC housing. If the PIC is connected via the ICSP-connector, then it needs to know this (but not the size of the housing). The parameter of the --SOCKET option is used give usburn this information.
Possible parameters for a PIC in the testsocket are **8**, **14**, **18, 20**, **28** and **40**. These are the number of pins of the DIL-PIC-housing.
If the PIC is connected via ICSP, then the parameter **ICSP** has to be used.

Don't forget:

The test socket of Brenner8 can only be used for 14-bit-core-PICs (PIC16F…
and some PIC12F…) and 16-bit-core-PICs (PIC18F…). all other PICs have to
be connected via ICSP-connector.
If a PIC is plugged into the test socked, then Pin 1 of the PIC has to be
plugged into Pin 1 of the 40-pin-testsocket.

By default usburn can program PICs via ICSP and 18-pin-PICs. If one likes to
program such PICs, then the --SOCKET option is not necessary.
Consequently programmers without testsocket (Brenner8mini, Brenner9) don't
need the -- SOCKET option at all.

If the --FAMILY option is used with the parameter 18J, 24, 30 or 33 then the
programmer selects ICSP automatically. A --SOCKET option would be
ignored.
Example:
```
    usburn --write  --SOCKET ICSP  -F18 --IN test.hex
    usburn -w  -S28 -F16 -Itest.hex
```


-F   --FAMILY   PIC-family (core architecture)
There are several different PIC-families on the marked. Usburn needs
information about the family membership of the target PIC. This is done by the
parameter of the --FAMILY option. The following parameters are possible:
10, **16**, **18**, **18J**, **18K**, **24**, **30**, **33**

Parameter	Description	Types
10	All 12-bit-core-PICs	all PIC10Fxxx all PIC1xF5x all PIC1xF5xx
16	All 14-bit-core-PICs	all PIC16Fyx except y=5 all PIC16Fxxx all PIC12Fyxx except y=5
18	16-bit-core-PICs	all PIC18Fxxx all PIX18Fxxxx
18J	16-bit-core-PICs	All PIC18FxxJxx
18K	16-bit-core-PICs	All PIC18FxxKxx
24	All PIC24- types	All PIC24....
30	All dsPIC30F-types	All dsPIC30F....
33	All dsPIC33F- types	all dsPIC33F....

The 12-bit-core-PICs require the additional --PIC option with the correct PIC-
name as parameter.
Example:
```
    usburn --write  --SOCKET ICSP  -F18 --IN test.hex
    usburn -w  --SOCKET ICSP  -F10 -PPIC10F206 --IN test.hex
    usburn -w  -S28 -F16 -Itest.hex
```


-H   --HEX   name of HEX-file

Usburn can read a HEX-file (for --write and --compare) and write a HEX-file (for –read). The Default file-names are
- HexIn.hex for Input
- HexOut.hex for Output

This option defines a new filename that is used as input-file-name and as output-file-name
Example:
```
usburn --write  --SOCKET ICSP  -F18 --HEX test.hex
usburn -w  -S28 -F16 -Htest.hex
usburn --read  --SOCKET ICSP  -F18 --HEX  test.hex
usburn -r  -S28 -F16 -Htest.hex
```

-I   --IN   name of input HEX-file (for write & compare)
Usburn can read a HEX-file (for --write and --compare). The default file name is „**HexIn.hex**".
This option can define a different file name for the input HEX-file.
Example:
```
usburn --write  --SOCKET ICSP  -F18 --IN test.hex
usburn -w  -S28 -F16 -Itest.hex
```

-L   --OSCCAL   Osccal-value for oscillator finetuning
Some PICs have an internal oscillator that can be fine-tuned by a number (OSCCAL value). The manufacture determines the optimum correction value and stores it inside the Flash-memory of the PIC.
During programming (option --write) usburn reads this value and re-burns it into the Flash memory. Thus the value doesn't get lost.
If a target PIC contains a OSCCAL-value, then usburn shows this value on the console.
In the OSCCAL value got lost (you know: shit happens) or was changed for a reason, then a new OSCCAL value can be programmed into the PIC with the --OSCCAL option. Possible parameters are
- For 12-bit-core PICs : -64 .. +63
- For 14-bit-core-PICs : 0 … 63

Other values would be ignored.
Example:
```
usburn --write  -S18  -F16 -L32 --IN test.hex
usburn -w  -SICSP -F10PIC10F206 -L-12 -Itest.hex
```

-O   --OUT   name of output HEX-file (for read)
Usburn can write a HEX-file (for --read). The default file name is „**HexOut.hex**".
This option can define a different file name for the output HEX-file.
Example:
```
usburn --read  --SOCKET ICSP  -F18 --OUT test.hex
usburn -r  -S28 -F16 -Otest.hex
```

-P   --PIC   manual selection of the PIC-type

To work correctly usburn needs to know the correct PIC-type. This is normally detected automatically, but there are two situations, where the automatic type identification can not function:

- Reanimation of a deaf PIC (--reanimate)
- Programming of 12-bit core PICs (--FAMILY 10)

In this situations the **--PIC** option has to be used, to hand over the exact PIIT-type to usburn. The parameter for the --PIC option is the exact name of the PIC in upper cases without (housing specific) extensions. The option --list can be used to get a list of all supported PIC-names in the correct writing. If the name of a low-power PIC (with L in the middle of the name) is not in the list, then the name without the L has to be used.

For a „Pic16LF628A-04/P" the correct parameter would be „PIC16F628A".
Example:

```
usburn --reanimate  --PIC PIC18F2450
usburn --w  --SOCKET ICSP  -F10  -PPIC10F202
```

1.5   To program a target PIC

To burn a HEX-file into a target PIC usburn needs the following information:

- The burn-option: --write
- Information about the connection between programmer and PIC: --SOCKET
- The family of the PIC: --FAMILY
- The name of the HEX-file: --IN

Now some examples with long and short options.

To program a PIC16F876 (28 Pins) that sits in the test socket of the Brenner8 with the HEX-file blink.hex :

```
usburn --write  --SOCKET 28  -FAMILY 16 --IN blink.hex
usburn -w  -S28 -F16 -Iblink.hex
```

To program a PIC18F2450 via ICSP-connector with the HEX-file test.hex :

```
usburn --write  --SOCKET ICSP  - FAMILY 18 --IN test.hex
usburn -w  -SICSP -F18 -Itest.hex
```

To program a PIC10F202 via ICSP-connector with the HEX-file test.hex:

```
usburn --write  --SOCKET ICSP  - FAMILY 10 --PIC /
     PIC10F202 --IN test.hex
usburn -w  -SICSP -F10 -PPIC10F202 -Itest.hex
```

To read out a PIC16F628 (18 Pins) that sits in the test socket and to write the data into the file result.hex :

```
usburn --read  --SOCKET 18  -FAMILY 16 --OUT result.hex
usburn -r  -S18 -F16 -Oresult.hex
```

To check, if a PIC16F876 (28 Pins in testsocket) contains the same data like the HEX-file blink.hex :

```
usburn --compare  --SOCKET 28  -FAMILY 16 --IN blink.hex
usburn -c  -S28 -F16 -Iblink.hex
```

To upload the firmware-file fw_14.hex into the programmer:

```
usburn --firmware --IN fw_14.hex
usburn -f --IN fw_14.hex
```

2 Bootloader

From time to time new firmware is published for the Brenner8/9. That is necessary to fix bugs and to integrate new features and capabilities.
By the help of the Bootloader the new firmware can be easily transferred into the programmer.
The Bootloader is also the preferred tool to burn the very first firmware into a new programmer.

The Bootloader is a little piece of software, which resides in a special memory-part of the programmers control PIC. It has to be burned into the control-PIC (PIC18F2550) by the help of a different programmer (hen-egg-problem). The Bootloader for the Brenner8/9 is the "Bootloader_0". It is contained in the usburn-TAR-file. There are different versions of the Bootloader in the TAR-file, designed for the use with different clock-speeds. If your programmer is using the default 20-MHz-crystal, then you have to use **boot_0_20mhz.hex**.

From now I assume, that the Bootloader was burned into the programmers control PIC.

2.1   Use of the Bootloader with usburn (Linux)

For normal operation of Brenner8/9 the Bootloader is not needed, it is inactive.
But if a (new) firmware has to be burned into the programmer, then the Bootloader has to be activated. There are two ways to do this:
* activate the bootloader with usburn (preferred way)
* activate the bootloader with jumper JP1 (backup-method)

2.1.1 Use of the Bootloader with usburn

To load a new firmware into the programmer usburn can be used with the --firmware option. In addition the --IN option has to be used to define the name of the firmware-HEX-file.
Example:

* `usburn --firmware --IN fw_14.hex`
* `usburn -f -Ifw_14.hex`

Usburn will now detect the programmer, switch it in Bootloader-mode and reset it. It will now be disconnected and again connected to the PC (this is done by software), and both LEDs of the programmer will light up to indicate bootloader-mode. This need 3...4 seconds.
Now the defined HEX-file (contains the firmware) will be opened and the firmware will be burned into the programmer. The new firmware will be checked for correctness. If this was successful, then the programmer is

switched back into normal mode and usburn quits. Both LEDs flash a single time.

Ready.