Package HWSUPP - Hardware support
Version 3.10.4

The fli4l-Team
email: team@fli4l.de

October 25, 2015

Contents

1. Documentation of the HWSUPP package

1.1. HWSUPP - Hardware support

1.1.1. Description

This package supplies the support for special hardware components.
Supported are:

- Temperature sensors
- LEDs
- Voltage sensors
- Fan speed
- Buttons
- Watchdog
- VPN cards

The following systems, mainboards and VPN cards are supported:

- Standard PC hardware
  - PC keyboard LEDs
- ACPI Hardware
- Embedded systems
  - AEWIN SCB6971
  - Fujitsu Siemens Futro S200
  - PC Engines ALIX
  - PC Engines APU
  - PC Engines WRAP
  - Soekris net4801
  - Soekris net5501
- Mainboards
  - Commell LE-575
  - GigaByte GA-M521-S3
  - LEX CV860A

  - SuperMicro PDSME
  - SuperMicro X7SLA
  - Tyan S5112
  - WinNet PC640
  - WinNet PC680

- VPN cards (PCI, miniPCI and miniPCIe)
  - vpn1401 vpn1411

1.1.2. Configuration of the HWSUPP package

The configuration is made, as for all fli4l packages, by adjusting the file `path/fli4l-3.10.4/<config>/hwsupp.txt` to meet your own demands.

OPT_HWSUPP The setting `'no'` deactivates the OPT_HWSUPP package completely. There will be no changes made to the fli4l boot medium or the archive `opt.img`. OPT_HWSUPP does not overwrite any other parts of the fli4l installation. To activate OPT_HWSUPP set the variable `OPT_HWSUPP` to `'yes'`.

HWSUPP_TYPE This configuration variable sets the type of supported hardware. Following values can be used:

- sim
- generic-pc
- generic-acpi
- aewin-scb6971
- commell-le575
- fsc-futro-s200
- gigabyte-ga-m52l-s3
- lex-cv860a
- pcengines-alix
- pcengines-apu
- pcengines-wrap
- soekris-net4801
- soekris-net5501
- supermicro-pdsme
- supermicro-x7sla
- tyan-s5112
- winnet-pc640
- winnet-pc680

HWSUPP_WATCHDOG The setting `'yes'` activates the watchdog daemon if the hardware contains a watchdog. The watchdog will automatically restart a non responding system.

HWSUPP_CPUFREQ The setting `'yes'` activates CPU frequency adjustment controls.

HWSUPP_CPUFREQ_GOVERNOR Selection of CPU frequency governor. The selected governor controls the frequency adjustment behaviour. It's a selection of one of:

- performance
  The CPU allways runs with the highest available frequency.
- ondemand
  The CPU frequency will be adjusted depending on the current CPU usage. The frequency can change very quickly.
- conservative
  The CPU frequency will be adjusted depending on the current CPU usage. The frequency is changed step by step.
- powersave
  The CPU allways runs with the lowest available frequency.
- userspace
  The CPU frequence kann be set manually or by an user script via the sysfs variable `/devices/system/cpu/cpu`<*n*>`/cpufreq/scaling_setspeed`.

HWSUPP_LED_N Defines the number of LEDs. The number of LEDs of the hardware in use should be entered here.

HWSUPP_LED_x Defines the information indicated by the LED. The following informations are possible:

- ready - the fli4l router ist ready for operation[1]
- online - the fli4l router has an active internet connection
- trigger - LED is controlled by a kernel trigger
- user - LED is controlled by an user script

The list of possible indications can be extended by other packages. For example, if the WLAN package is loaded the information

- wlan - WLAN is active

is possible.

In apppendix A.4 package developers can get some hints on how to create such extensions.

HWSUPP_LED_x_DEVICE Specifies the LED device.

Here you either have to enter a LED device (to be found at `/sys/class/leds/` in the router's file system) or a GPIO[2]number.

[1] If `HWSUPP_LED_x='ready'` is set, the boot progress is indicated by a blink sequence (see appendix A.3).
[2] A GPIO (General Purpose Input/Output) is a generic pin on an integrated circuit whose behavior can be programmed at run time, including whether it is an input or output pin.

A list of valid LED device names for a specific HSUPP_TYPE can be found in the appendix A.1.1.

The GPIO number has to be enterd in the format gpio::x. If a GPIO is entered, the corresponding LED device will be created automatically. By preceding the char / the GPIO functionality may be inverted.

Examples:

```
HWSUPP_LED_1_DEVICE='apu::1'      # LED 1 on PC engines APU
HWSUPP_LED_2_DEVICE='gpio::237'   # GPIO 237
HWSUPP_LED_3_DEVICE='/gpio::245'  # inverted GPIO 245
```

HWSUPP_LED_x_PARAM Defines parameters for the selected LED information.

Depending on the selection in in HWSUPP_LED_x, in HWSUPP_LED_x_PARAM different settings are possible.

If HWSUPP_LED_x='trigger' is set, the trigger name has to be specified in HWSUPP_LED_x_PARAM.

Available triggers can be displayed with the shell command cat /sys/class/leds/*/trigger.

Besides triggers created by e.g. netfilter or hardware drivers like ath9k, further trigger modules can be loaded via HWSUPP_DRIVER_x.

Examples:

```
HWSUPP_LED_1='trigger'
HWSUPP_LED_1_TRIGGER='heartbeat'
HWSUPP_LED_2='trigger'
HWSUPP_LED_2_TRIGGER='netfilter-ssh'
```

If 'HWSUPP_LED_x' has the value 'user' in HWSUPP_LED_PARAM a valid script name including path has to be entered.

Example:

```
HWSUPP_LED_1='user'
HWSUPP_LED_1_PARAM='/usr/local/bin/myledscript'
```

When HWSUPP_LED_x='wlan' is set, the WLAN devices have to be entered in HWSUPP_LED_x_PARAM.

Defines one ore more WLAN devices, whose state shall be displayed. Multiple WLAN devices have to be separated by spaces.

When the state of multiple WLAN devices should be indicated by a single LED, the LED has the following meaning:

- off - all WLAN devices are inactiv
- blinking - some WLAN device(s) is/are active
- on - all WLAN devices are active

Example:

```
HWSUPP_LED_1='wlan'
HWSUPP_LED_1_WLAN='wlan0 wlan1'
```

HWSUPP_BOOT_LED Definies a LED to indicate the boot progress by a blink sequence.

When `HWSUPP_LED_x='ready'` is set for any LED, this setting is used and `HWSUPP_BOOT_LED` will be ignored.

HWSUPP_BUTTON_N Defines the number of buttons.
The number of buttons of the hardware in use should be entered here.

HWSUPP_BUTTON_x Defines the action which should be executed on button press.
The following actions are supported:

- reset - restart the fli4l router
- online - causes an internet dialin or terminates an internet connection.
- user - an user script will be executed

Th e list of possible actions can be extended by other packages. If the WLAN package is loaded, eg. the action

- wlan - activate or deactivate WLAN

is possible.

HWSUPP_BUTTON_x_DEVICE Specifies the button device an.

Here has to be entered a GPIO number in the format `gpio::x`. By preceding the char `/` the GPIO functionality may be inverted.

A list of predefined GPIO's for a specific `HSUPP_TYPE` can be found in the appendix A.1.2.

Examples:

```
HWSUPP_BUTTON_1_DEVICE='gpio::252'
HWSUPP_BUTTON_2_DEVICE='/gpio::237'
```

HWSUPP_BUTTON_x_PARAM Defines parameters for the action selected in `HWSUPP_BUTTON_x`.

Depending on the action `HWSUPP_BUTTON_x_PARAM` has different meanings.

If `HWSUPP_BUTTON_x='user'` is set, `HWSUPP_BUTTON_x_PARAM` defines a script to be executed on button press.

Example:

```
HWSUPP_BUTTON_1='user'
HWSUPP_BUTTON_2_WLAN='/usr/local/bin/myscript'
```

If `HWSUPP_BUTTON_x` is set to `'wlan'`, the `HWSUPP_BUTTON_x_PARAM` defines one ore more WLAN devices, which shall be activated or deactivated on button press. Multiple WLAN devices have to be separated by spaces.

Example:

```
HWSUPP_BUTTON_2='wlan'
HWSUPP_BUTTON_2_WLAN='wlan0 wlan1'
```

1.1.3. Expert settings

The following settings should only be touched if you know exactly

- which hardware you have,
- which additional drivers it needs and
- the addresses and types of I^2C[3]devices.

Activating the expert settings will issue a warning during the mkfli4l build.

HWSUPP_DRIVER_N Number of additional drivers. The drivers in `HWSUPP_DRIVER_x` will be loaded in the denoted order.

HWSUPP_DRIVER_x Driver name (without file extension `.ko`).

Example:

```
HWSUPP_DRIVER_N='2'
HWSUPP_DRIVER_1='i2c-piix4'     # I2C bus driver
HWSUPP_DRIVER_2='gpio-pcf857x'  # I2C GPIO expander
```

HWSUPP_I2C_N Number of I^2C devices to be loaded.

I^2C doesn't support any PnP mechanismn. Hence for each I^2C device the bus number, the device address and the device type have to be specified.

HWSUPP_I2C_x_BUS I^2C bus number the device is attached to.

The bus number has to be entered as `i2c-x`.

HWSUPP_I2C_x_ADDRESS The device's I^2C address.

The address has to be entered as a hex number in the range between `0x03` and `0x77`.

HWSUPP_I2C_x_DEVICE The type of I^2C device which is supported by an already loaded driver.

Example:

```
HWSUPP_I2C_N='1'
HWSUPP_I2C_1_BUS='i2c-1'
HWSUPP_I2C_1_ADDRESS='0x38'
HWSUPP_I2C_1_DEVICE='pcf8574a' # supported by gpio-pcf857x driver
```

[3] An I^2C bus or SMBus is a serial bus used in PCs eg. to read temperature sensor values. In many cases an I^2C bus or SMBus is available on a pin header and can be used for own hardware extensions.

1.1.4. Support for VPN cards

OPT_VPN_CARD The setting `'no'` deactivates the OPT_VPN_CARD package completely. There will be no changes made to the fli4l boot mediums or the archive `opt.img`. OPT_VPN_CARD does not overwrite any other parts of the fli4l installation. To activate OPT_VPN_CARD set the variable `OPT_VPN_CARD` to `'yes'`.

VPN_CARD_TYPE This configuration variable defines the type of the VPN accelerator. The following values are supported:

- hifn7751 - Soekris vpn1401 and vpn1411
- hifnhipp

A. Appendix to the HWSUPP package

A.1. HWSUPP - Device dependant settings

A.1.1. Available LED devices

Depending on the `HWSUPP_TYPE` different LED devices are present. For hardware not listed here the PC keyboard LEDs are available using generic-pc.

Additional LED devices can be mounted on eg. WLAN adapters. The valid LED device names can be determined by executing `ls /sys/class/leds/`, eg. via ssh on the router's console.

sim

LED simulation, will log to `syslog`:

- `simu::1`
- ...
- `simu::8`

generic-pc

PC keyboard LEDs:

- `keyboard::scroll`
- `keyboard::caps`
- `keyboard::num`

generic-acpi

PC keyboard LEDs, like generic-pc

pcengines-alix

- `alix::1`
- `alix::2`
- `alix::3`

pcengines-apu

- `apu::1`
- `apu::2`
- `apu::3`

pcengines-wrap

- `wrap::1`
- `wrap::2`
- `wrap::3`

soekris-net4801

- `net48xx::error`

soekris-net5501

- `net5501::error`

A.1.2. Available Button Devices

Depending on the `HWSUPP_TYPE` different GPIO devices are predefined for buttons.

pcengines-alix

- `gpio::24`

pcengines-apu

- `gpio::252`

pcengines-wrap

- `gpio::40`

soekris-net5501

- `gpio::25`
  The button is named 'Reset' on the soekris case.
  Attention: the button must be enabled in BIOS.

A.1.3. Hardware specific notes

pcengines-alix

A faulty driver for the lm90 temperatur sensor causes a loss of temperature monitoring.

As a workaraound the lm90 driver will be unloaded and reloaded again automatically by a cron job. This requires the package easycron to be loaded (`OPT_EASYCRON='yes'`).

A.2. HWSUPP - Configuration examples

A.2.1. generic-pc

```
OPT_HWSUPP='yes'
HWSUPP_TYPE='generic-pc'

HWSUPP_WATCHDOG='no'
HWSUPP_CPUFREQ='no'

HWSUPP_LED_N='3'
HWSUPP_LED_1='ready'
HWSUPP_LED_1_DEVICE='keyboard::num'
HWSUPP_LED_2='online'
HWSUPP_LED_2_DEVICE='keyboard::caps'
HWSUPP_LED_3='wlan'
HWSUPP_LED_3_DEVICE='keyboard::scroll'
HWSUPP_LED_3_WLAN='wlan0'

HWSUPP_BUTTON_N='0'
```

A.2.2. pcengines-apu

```
OPT_HWSUPP='yes'
HWSUPP_TYPE='pcengines-apu'

HWSUPP_WATCHDOG='yes'
HWSUPP_CPUFREQ='yes'
HWSUPP_CPUFREQ_GOVERNOR='ondemand'

HWSUPP_LED_N='3'
HWSUPP_LED_1='ready'
HWSUPP_LED_1_DEVICE='apu::1'
HWSUPP_LED_2='wlan'
HWSUPP_LED_2_DEVICE='apu::2'
HWSUPP_LED_2_WLAN='wlan0'
HWSUPP_LED_3='online'
HWSUPP_LED_3_DEVICE='apu::3'

HWSUPP_BUTTON_N='1'
HWSUPP_BUTTON_1='wlan'
HWSUPP_BUTTON_1_DEVICE='gpio::252'
HWSUPP_BUTTON_1_PARAM='wlan0'
```

A.2.3. pcengines-apu with GPIO's

```
OPT_HWSUPP='yes'
```

```
HWSUPP_TYPE='pcengines-apu'

HWSUPP_WATCHDOG='yes'
HWSUPP_CPUFREQ='yes'
HWSUPP_CPUFREQ_GOVERNOR='ondemand'

HWSUPP_LED_N='5'
HWSUPP_LED_1='ready'
HWSUPP_LED_1_DEVICE='apu::1'
HWSUPP_LED_2='wlan'
HWSUPP_LED_2_DEVICE='apu::2'
HWSUPP_LED_2_WLAN='wlan0'
HWSUPP_LED_3='online'
HWSUPP_LED_3_DEVICE='apu::3'
HWSUPP_LED_4='trigger'
HWSUPP_LED_4_PARAM='phy0rx'
HWSUPP_LED_4_DEVICE='gpio::237'
HWSUPP_LED_5='trigger'
HWSUPP_LED_5_PARAM='phy0tx'
HWSUPP_LED_5_DEVICE='gpio::245'

HWSUPP_BUTTON_N='2'
HWSUPP_BUTTON_1='wlan'
HWSUPP_BUTTON_1_DEVICE='gpio::252'
HWSUPP_BUTTON_1_PARAM='wlan0'
HWSUPP_BUTTON_2='online'
HWSUPP_BUTTON_2_DEVICE='gpio::236'
```

A.3. HWSUPP - Blink Sequences

The following blink sequences are displayed during boot:

1. ⊗ ⊗ ...
2. ⊗ ⊗ ⊗ ⊗ ...
3. ⊗ ⊗ ⊗ ⊗ ⊗ ⊗ ...
4. ⊗ ⊗ ⊗ ⊗ ⊗ ⊗ ⊗ ⊗ ...

The first sequence is displayed while processing `rc002.*` to `rc250.*`
(1 * blink - pause),
for `rc250.*` to `rc500.*` the second (2 * blink - pause),
for `rc500.*` to `rc750.*` the third and
for `rc750.*` until the end of the boot process the forth sequence (coninuous blinking).

A.4. HWSUPP - Hints for package developers

This chapter describes the things a package developer has to do to add LED or button functionality to a package[1].

A.4.1. LED extensions

LED type

Within the file `check/myopt.exp` the list of LED types which can be entered in `HWSUPP_LED_x` is extended.

Example:

```
+HWSUPP_LED_TYPE(OPT_MYOPT) = 'myopt'
                            : ', myopt'
```

Parameter check

Within `check/myopt.ext` the parameters which can be entered in `HWSUPP_LED_x_PARAM` will be checked.

Example:

```
if (opt_hwsupp)
then
    depends on hwsupp version 4.0

    foreach i in hwsupp_led_n
    do
        set action=hwsupp_led_%[i]
        set param=hwsupp_led_%_param[i]
        if (action == "myopt")
        then
            if (!(param =~ "(RE:MYOPT_LED_PARAM)"))
            then
                error "When HWSUPP_LED_\${i}='myopt', ...
                       must be entered in HWSUPP_LED_\${i}_PARAM"
            fi
        fi
    done
fi
```

LED Display

The command `/usr/bin/hwsupp_setled <LED> <status>/` has to be executed to set a LED in a package script (eg. `/usr/bin/<opt>_setled`)

The LED number can be found in `/var/run/hwsupp.conf`.

Status can be `off`, `on` or `blink`.

[1] Search for `##HWSUPP##` in the WLAN package to find the places to adapt.

Example:

```
if [ -f /var/run/hwsupp.conf ]
then
    . /var/run/hwsupp.conf
    [ 0$hwsupp_led_n -eq 0 ] || for i in `seq 1 $hwsupp_led_n`
    do
        eval action=\$hwsupp_led_${i}
        eval param=\$hwsupp_led_${i}_param
        if [ "$action" = "<opt>" ]
        then
            if [ <myexpression> ]
            then
                /usr/bin/hwsupp_setled $i on
            else
                /usr/bin/hwsupp_setled $i off
            fi
        fi
    done
fi
```

The actual state of a LED can be queried with `/usr/bin/hwsupp_getled <LED>/`. The result will be `off`, `on` or `blink`.

A.4.2. Button extensions

A.4.3. Button action

In `check/myopt.exp` the list of button types allowed in `HWSUPP_LED_x` can be extended.

Beispiel:

```
+HWSUPP_BUTTON_TYPE(OPT_MYOPT) = 'myopt'
                               : ', myopt'
```

Parameter check

The parameters which can be entered in `HWSUPP_BUTTON_x_PARAM` will be checked using `check/myopt.ext` .

Example:

```
if (opt_hwsupp)
then
    depends on hwsupp version 4.0

    foreach i in hwsupp_button_n
    do
        set action=hwsupp_buttonn_%[i]
        set param=hwsupp_button_%_param[i]
        if (action == "myopt")
```

```
        then
            add_to_opt "files/usr/bin/myopt_keyprog" "mode=555 flags=sh"
            if (!(param =~ "(RE:MYOPT_BUTTON_PARAM)"))
            then
                error "When HWSUPP_BUTTON_\${i}='myopt', ...
                       must be entered in  HWSUPP_BUTTON_\${i}_PARAM"
            fi
        fi
    done
fi
```

Button function

When a button is pressed the script file `/usr/bin/myopt_keyprog` will be executed.
The content of `HWSUPP_BUTTON_x_PARAM` is passed as a parameter.
Example:

```
##TODO## example
```

Index