Paket HWSUPP - Unterstützung von Hardware
Version 3.10.5

Das fli4l-Team
E-Mail: team@fli4l.de

16. Februar 2016

Inhaltsverzeichnis

1. Dokumentation des Paketes HWSUPP

1.1. HWSUPP - Unterstützung von Hardware

1.1.1. Beschreibung

Das Paket stellt die Unterstützung für die Nutzung spezieller Hardwarekomponenten bereit.
Unterstützte Hardwarekomponenten/-elemente:

- Temeratursensoren
- LEDs
- Spannungssensoren
- Lüfterdrehzahlen
- Taster
- Watchdog
- VPN-Karten

Unterstützung gibt es für die folgende Systeme/Mainboards/VPN-Karten:

- Standard PC-Hardware
  - LEDs einer PC-Tastatur
- ACPI-PC-Hardware
- Embedded Systeme
  - 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 Karten (PCI, miniPCI and miniPCIe)
  - vpn1401 vpn1411

1.1.2. Konfiguration des Paketes HWSUPP

Die Konfiguration erfolgt, wie bei allen fli4l Paketen, durch Anpassung der Datei `Pfad/fli4l-3.10.5/<config>/hwsupp.txt` an die eigenen Anforderungen.

OPT_HWSUPP Die Einstellung `'no'` deaktiviert das OPT_HWSUPP vollständig. Es werden keine Änderungen am fli4l Archiv `rootfs.img` bzw. dem Archiv `opt.img` vorgenommen. Weiterhin überschreibt das OPT_HWSUPP grundsätzlich keine anderen Teile der fli4l Installation.
Um OPT_HWSUPP zu aktivieren, ist die Variable `OPT_HWSUPP` auf `'yes'` zu setzen.

HWSUPP_TYPE In dieser Konfigurationsvariable wird die zu unterstützende Hardware festgelegt. Folgende Werte stehen zur Verfügung:

- 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 Die Einstellung `'yes'` aktiviert den Watchdog-Daemon falls die gewählte Hardware einen Watchdog besitzt. Durch den Watchdog wird ein hängendes System automatisch neu gestartet werden.

HWSUPP_CPUFREQ Die Einstellung `'yes'` aktiviert die Anpassung der Prozessor-Taktfrequenz.

HWSUPP_CPUFREQ_GOVERNOR Auswahl des CPU-Frequenz-Reglers. Die Auswahl des Reglers steuert das Verhalten der Anpassung der Prozessor-Taktfrequenz. Zur Auswahl stehen:

- performance

  Der Prozessor läuft immer mit der maximalen Taktfrequenz.

- ondemand

  Die CPU-Frequenz wird an die Rechenleistung angepasst. Dabei kann die CPU-Frequenz u.U. sprunghaft angehoben oder abgesenkt werden.

- conservative

  Die CPU-Frequenz wird an die Rechenleistung angepasst. Die CPU-Frequenz wird schrittweise angehoben bzw. abgesenkt.

- powersave

  Der Prozessor läuft immer mit der minimmalen Taktfrequenz.

- userspace

  Der Prozessortakt kann manuell oder von einem Anwenderskript über die sysfs-Variable `/devices/system/cpu/cpu<n>/cpufreq/scaling_setspeed` gesetzt werden.

HWSUPP_LED_N Definiert die Anzahl der LEDs. Hier sollte die Anzahl der LEDs die die verwendete Hardware bereitstellt stehen.

HWSUPP_LED_x Definiert die Information, die durch das LED angezeigt werden soll. Folgende Informationen sind möglich:

- ready - Der fli4l-Router ist betriebsbereit[1]
- online - der fli4l-Router ist mit den Internet verbunden
- trigger - Anzeige wird durch einen LED-Trigger gesteuert
- user - Anzeige wird durch ein Benutzerscript gesteuert

Die Liste der möglichen Information kann durch andere Pakete erweitert werden. So ist bei geladenem WLAN-Paket z.B. die Anzeige

- wlan - das WLAN ist aktiviert

möglich.

Im Anhang A.4 finden sich Hinweise für Paket-Entwickler wie eine solche Erweiterung anzulegen ist.

[1] Ist `HWSUPP_LED_x='ready'`, so wird der Bootfortschritt durch eine Blink-folge angezeigt (siehe Anhang A.3).

HWSUPP_LED_x_DEVICE Gibt das LED-Device an.

Hier wird entweder ein LED-Device eingetragen (zu finden unter `/sys/class/leds/` im Dateisystem des Routers) oder eine GPIO[2]-Nummer.

Eine Liste gültiger Namen der LED-Devices für den jeweiligen `HSUPP_TYPE` findet sich im Anhang A.1.1.

Die GPIO-Nummer musss im Format `gpio::x` eingegeben werden. Wenn man ein GPIO eingeträgt, so wird das dazugehörige LED-Device automatisch angelegt. Durch Voranstellen von `/` wird die Funktionsweise des GPIO invertiert.

Beispiele:

```
HWSUPP_LED_1_DEVICE='apu::1'
HWSUPP_LED_2_DEVICE='gpio::237'
HWSUPP_LED_3_DEVICE='/gpio::245'
```

HWSUPP_LED_PARAM Definiert Parameter für die ausgewählte LED Anzeige.

Je nach Auswahl in `HWSUPP_LED_x` hat `HWSUPP_LED_x_PARAM` eine unterschiedliche Bedeutung.

Ist `HWSUPP_LED_x='trigger'`, so ist der Name des LED-Triggers, der die Ansteuerung der LED kontrolliert, in `HWSUPP_LED_x_PARAM` einzutragen.

Die verfügbaren Trigger können mit dem Shell-Kommando `cat /sys/class/leds/*/trigger` angezeigt werden.

Neben den Triggern die von z.B. netfilter oder Hardwaretreibern wie ath9k erzeugt werden, können weitere Trigger-Module über `HWSUPP_DRIVER_x` geladen werden.

Beispiele:

```
HWSUPP_LED_1='trigger'
HWSUPP_LED_1_PARAM='heartbeat'
HWSUPP_LED_2='trigger'
HWSUPP_LED_2_PARAM='netfilter-ssh'
```

Ist `HWSUPP_LED_x='user'`, so ist in `HWSUPP_LED_PARAM` ein gültiger Skriptname inclusive Pfad einzutragen.

Beispiel:

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

[2]Ein GPIO (General Purpose Input/Output) ist ein Kontaktstift an einem integrierten Schaltkreis, dessen Verhalten durch Programmierung bestimmbar ist, unabhängig, ob als Ein- oder Ausgabekontakt.

Ist `HWSUPP_LED_x='wlan'`, so definiert `HWSUPP_LED_PARAM` ein oder mehrere WLAN Devices, deren Zustand angezeigt wird. Mehrere WLAN Devices sind durch Leerzeichen zu trennen.

Wird der Zustand mehrerer WLAN Devices mit einer LED Angezeigt, so hat die LED folgende Bedeutung:

- aus - alle WLAN Devices sind inaktiv
- blinkt - ein Teil der WLAN Devices ist aktiv
- an - alle WLAN Devices sind aktiv

Beispiel:

```
HWSUPP_LED_1='wlan'
HWSUPP_LED_1_PARAM='wlan0 wlan1'
```

HWSUPP_BOOT_LED Definiert eine LED die während des Bootvorgangs den Fortschritt durch eine Blinkfolge angezeigt.

Wenn für eine LED `HWSUPP_LED_x='ready'` gesetzt ist so hat diese Einstellung Vorrang und `HWSUPP_BOOT_LED` wird ignoriert.

HWSUPP_BUTTON_N Definiert die Anzahl der BUTTONs. Hier sollte die Anzahl der Taster die die verwendete Hardware bereitstellt stehen.

HWSUPP_BUTTON_x Definiert die Aktion die durch drücken des Tasters durchgeführt werden soll. Folgende Aktionen sind möglich:

- reset - Startet den fli4l-Router neu
- online - Baut die Internetverbindung auf bzw. beendet diese
- user - Ein User-Script wird ausgeführt

Die Liste der möglichen Aktionen kann durch andere Pakete erweitert werden. So ist bei geladenem WLAN-Paket z.B. die Aktion

- wlan - WLAN aktivieren bzw. deaktivieren

möglich.

HWSUPP_BUTTON_x_DEVICE Gibt das Button-Device an. Hier ist eine GPIO-Nummer einzutragen.

Die GPIO-Nummer musss im Format `gpio::x` eingegeben werden. Durch Voranstellen von `/` wird die Funktionsweise des GPIO invertiert.

Eine Liste der vordefinierten GPIO's für den jeweiligen `HSUPP_TYPE` findet sich im Anhang A.1.2.

Beispiele:

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

HWSUPP_BUTTON_x_PARAM Definiert Parameter für die ausgewählte Aktion.

Je nach Wert in `HWSUPP_BUTTON_x` hat `HWSUPP_BUTTON_x_PARAM` eine unterschiedliche Funktion.

Ist `HWSUPP_BUTTON_x='user'`, so definiert `HWSUPP_BUTTON_x_PARAM` ein Script das beim Drücken des Tasters ausgeführt werden soll.

Beispiel:

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

Ist `HWSUPP_BUTTON_x_ACTION='wlan'`, so sind in `HWSUPP_BUTTON_x_PARAM` das oder die WLAN Devices einzutragen, die durch Drücken des Tasters aktiviert bzw. deaktiviert werden. Mehrere WLAN Devices sind durch Leerzeichen zu trennen.

Beispiel:

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

1.1.3. Experten-Einstellungen

Die folgenden Einstellungen sollten nur gemacht werden, wenn man genau weiss

- welche Hardware man hat und welche zusätzlichen Treiber man dafür benötigt
- an welchen Adressen welche I²C-Geräte[3]liegen.

Nach dem Aktivieren der Experteneinstellungen erhält man eine Warnung beim mkfli4l Bau.

HWSUPP_DRIVER_N Anzahl der zusätzlich zu ladenden Treiber. Die Treiber in `HWSUPP_DRIVER_x` werden in der angegebenen Reihenfolge geladen.

HWSUPP_DRIVER_x Name des Treibers (ohne Dateiendung `.ko`).

Beispiel:

```
HWSUPP_DRIVER_N='2'
HWSUPP_DRIVER_1='i2c-piix4'     # I2C Bus Treiber
HWSUPP_DRIVER_2='gpio-pcf857x'  # I2C GPIO Expander
```

HWSUPP_I2C_N Anzahl der zu ladenden I²C-Geräte.

I²C unterstützt keine PnP-Mechanismen. Daher sind für jedes zu ladende I²C-Gerät die Busnummer, die Geräteadresse und der Gerätetyp anzugeben.

[3]Ein I²C-Bus oder SMBus ist ein serieller Bus der im PC z.B. zum Auslesen von Temperatur-Sensoren verwendet wird. Vielfach ist der I²C-Bus oder SMBus auf einer Stiftleiste verfügbar und kann für eigene Hardwareerweiterungen genutzt werden.

HWSUPP_I2C_x_BUS I²C-Busnummer an der das zu ladende Gerät angeschlossen ist.

Die Bussnummer ist im Format `i2c-x` anzugeben.

HWSUPP_I2C_x_ADDRESS I²C-Busadresse des Geräts.

Die Adresse ist als Hexadezimalzahl im Bereich von `0x03` bis `0x77` anzugeben.

HWSUPP_I2C_x_DEVICE Der Typ des I²C-Geräts der vom einem zuvor geladenen Treiber erkannt wird.

Beispiel:

```
HWSUPP_I2C_N='1'
HWSUPP_I2C_1_BUS='i2c-1'
HWSUPP_I2C_1_ADDRESS='0x38'
HWSUPP_I2C_1_DEVICE='pcf8574a' # Unterstützt von gpio-pcf857x Treiber
```

1.1.4. Unterstützung von VPN-Karten

OPT_VPN_CARD Die Einstellung `'no'` deaktiviert das OPT_VPN_CARD vollständig. Es werden keine Änderungen am fli4l Archiv `rootfs.img` bzw. dem Archiv `opt.img` vorgenommen. Weiterhin überschreibt das OPT_VPN_CARD grundsätzlich keine anderen Teile der fli4l Installation.
Um OPT_VPN_CARD zu aktivieren, ist die Variable `OPT_VPN_CARD` auf `'yes'` zu setzen.

VPN_CARD_TYPE In dieser Konfigurationsvariable wird der zu unterstützende VPN Beschleuniger festgelegt. Folgende Werte stehen zur Verfügung:

- hifn7751 - Soekris vpn1401 und vpn1411
- hifnhipp

A. Anhang zum Paket HWSUPP

A.1. HWSUPP - Geräteabhängige Einstellungen

A.1.1. Verfügbare LED-Devices

Je nach `HWSUPP_TYPE` sind verschiedene LED-Devices verfügbar. Bei nicht aufgeführter Hardware sind die PC-Tastatur LED's wie bei generic-pc verfügbar.

Zusätzlich LED-Devices können z.B. auf WLAN-Karten verfügbar sein. Die gültigen Namen der LED-Devices ermittelt man mittels Eingabe von `ls /sys/class/leds/` z.B. per ssh auf der Router-Console.

sim

LED Simulation, erzeugt Eintrag im `syslog`:

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

generic-pc

PC-Tastatur LED's:

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

generic-acpi

PC-Tastatur-LED's, wie 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. Verfügbare Button-Devices

Je nach `HWSUPP_TYPE` sind verschiedene GPIO-Devices für Taste vorbelegt.

pcengines-alix

- `gpio::24`

pcengines-apu

- `gpio::252`

pcengines-wrap

- `gpio::40`

soekris-net5501

- `gpio::25`
  Der Taster ist am soekris Gehäuse mit 'Reset' beschriftet.
  Achtung: der Taster muss im BIOS freigeschaltet werden.

A.1.3. Hinweise zu spezieller Hardware

pcengines-alix

Beim Alix führt ein fehlerhafter Treiber für den lm90 Temperatursensor nach einiger Zeit zum Ausfall der Temperaturanzeige.

Als Workaraound wird der lm90 Treiber entladen und wieder neu geladen. Dies geschieht automatisch per cron-Job. Dazu muss das Paket easycron geladen werden (`OPT_EASYCRON='yes'`).

A.2. HWSUPP - Konfigurations-Beispiele

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 mit 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 - Blinkfolge

Die folgenden Blinkfolgen werden während des Bootvorgangs angezeigt:

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

Während der Abarbeitung von `rc002.*` bis `rc250.*` wird die erste Folge angezeigt
(1 * Blinken - Pause),
von `rc250.*` bis `rc500.*` die zweite (2 * Blinken - Pause),
von `rc500.*` bis `rc750.*` die 3. und
von `rc750.*` bis zum Ende des Bootvorgangs die 4. Folge (Dauerblinken).

A.4. HWSUPP - Hinweise für Paket-Entwickler

Im folgenden ist beschrieben was ein Paket-Entwickler zu tun hat, um Button- oder LED-Funktionalität zu einem Paket hinzuzufügen[1].

A.4.1. LED-Erweiterungen

LED-Typ

In der Datei `check/myopt.exp` wird die Liste der erlaubten LED-Typen die in `HWSUPP_LED_x` eingetragen werden können erweitert.

Beispiel:

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

Parameterprüfung

In der Datei `check/myopt.ext` werden die Parameter die für den neuen LED-Typen in `HWSUPP_LED_x_PARAM` eingetragen werden können geprüft.

Beispiel:

```
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
```

[1] Wenn man im WLAN-Paket nach `##HWSUPP##` sucht so findet man die anzupassenden Stellen.

LED schalten

Um eine LED zu schalten ist in einem eigenen Skript (z.B. `/usr/bin/<opt>_setled`) das Kommando `/usr/bin/hwsupp_setled <LED> <Status>/` aufzurufen.

Die LED-Nummer kann aus `/var/run/hwsupp.conf>` ausgelesen werden.

Als Status ist `off`, `on` oder `blink` zu übergeben.

Beispiel:

```
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
```

Den aktuellen Zustand einer LED kann man mit `/usr/bin/hwsupp_getled <LED>/` abfragen. Es wird je nach Status `off`, `on` oder `blink` ausgegeben.

A.4.2. Button-Erweiterungen

A.4.3. Button-Aktion

In der Datei `check/myopt.exp` wird die Liste der erlaubten Button-Typen die in `HWSUPP_BUTTON_x` eingetragen werden können erweitert.

Beispiel:

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

Parameterprüfung

In der Datei `check/myopt.ext` werden die Parameter, die für den neuen Button-Typen in `HWSUPP_BUTTON_x_PARAM` eingetragen werden können, geprüft.

Beispiel:

```
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-Funktion

Wenn eine Taste gedrückt wird, wird die Datei `/usr/bin/myopt_keyprog` ausgeführt.
Als Parameter wird er Inhalt von `HWSUPP_BUTTON_x_PARAM` übergeben
Beispiel:

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

Index