

## [PATCH] 2.6.0-test3 – Watchdog patches (sbc60xxwdt.c and advantechwdt.c)

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2003-08/2475.html>

---

*From:* Wim Van Sebroeck ([wim\\_at\\_iguana.be](mailto:wim_at_iguana.be))

*Date:* 08/09/03

Date: Sat, 9 Aug 2003 16:45:41 +0200  
To: [torvalds@osdl.org](mailto:torvalds@osdl.org)

Hi Linus,

please do a

bk pull <http://linux-watchdog.bkbits.net/linux-2.5-watchdog>

This will update the following files:

```
drivers/char/watchdog/advantechwdt.c | 17 –
drivers/char/watchdog/sbc60xxwdt.c | 332 ++++++-----
2 files changed, 209 insertions(+), 140 deletions(-)
```

through these ChangeSets:

<[wim@iguana.be](mailto:wim@iguana.be)> (03/08/09 1.1136)  
[WATCHDOG] sbc60xxwdt.c patch

- general cleanup of trailing spaces and comments
- fix possible wdt\_is\_open race
- add KERN\_\* to printk's
- changed watchdog\_info to correctly reflect what the driver offers
- added WDIOC\_GETSTATUS, WDIOC\_GETBOOTSTATUS, WDIOC\_SETTIMEOUT, WDIOC\_GETTIMEOUT, and WDIOC\_SETOPTIONS ioctls
- made timeout (the emulated heartbeat) a module\_param
- made the keepalive ping an internal subroutine
- added MODULE\_AUTHOR and MODULE\_DESCRIPTION info

<[wim@iguana.be](mailto:wim@iguana.be)> (03/08/09 1.1137)  
[WATCHDOG] sbc60xxwdt patch2

report default timeout as a number

<[wim@iguana.be](mailto:wim@iguana.be)> (03/08/09 1.1138)  
[WATCHDOG] sbc60xxwdt patch3

Linux-Kernel: [PATCH] 2.6.0-test3 – Watchdog patches (sbc60xxwdt.c and advantechwdt.c)

make wdt\_stop and wdt\_start module params

<wim@iguana.be> (03/08/09 1.1139)  
[WATCHDOG] sbc60xxwdt patch4

added extra printk's to report what problem occurred

<wim@iguana.be> (03/08/09 1.1140)  
[WATCHDOG] sbc60xxwdt.c patch5

some last clean-ups

<wim@iguana.be> (03/08/09 1.1141)  
[WATCHDOG] advantechwdt.c patch2

some small clean-ups (use PFX + report default timeout as it's value in the MODULE\_PARM\_DESC)

The ChangeSets can also be looked at on:

<http://linux-watchdog.bkbits.net:8080/linux-2.5-watchdog>

For completeness, I added the patches below.

Greetings,  
Wim.

---

```
diff -Nru a/drivers/char/watchdog/sbc60xxwdt.c b/drivers/char/watchdog/sbc60xxwdt.c
--- a/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:40:23 2003
+++ b/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:40:23 2003
@@ -7,51 +7,38 @@
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version
 * 2 of the License, or (at your option) any later version.
- *
- * The author does NOT admit liability nor provide warranty for
- * any of this software. This material is provided "AS-IS" in
- * the hope that it may be useful for others.
 *
- * (c) Copyright 2000 Jakob Oestergaard <jakob@ostenfeld.dk>
+ * The author does NOT admit liability nor provide warranty for
+ * any of this software. This material is provided "AS-IS" in
+ * the hope that it may be useful for others.
+ *
+ * (c) Copyright 2000 Jakob Oestergaard <jakob@unthought.net>
 *
 * 12/4 – 2000 [Initial revision]
 * 25/4 – 2000 Added /dev/watchdog support
 * 09/5 – 2001 [smj@oro.net] fixed fop_write to "return 1" on success
+ * 12/4 – 2002 [rob@osinvestor.com] eliminate fop_read
+ * fix possible wdt_is_open race
+ * add CONFIG_WATCHDOG_NOWAYOUT support
```

[PATCH] 2.6.0-test3 – Watchdog patches (sbc60xxwdt.c and advantechwdt.c)

```
+ * remove lock_kernel/unlock_kernel pairs
+ * added KERN_* to printk's
+ * got rid of extraneous comments
+ * changed watchdog_info to correctly reflect what the driver offers
+ * added WDIOC_GETSTATUS, WDIOC_GETBOOTSTATUS, WDIOC_SETTIMEOUT,
+ * WDIOC_GETTIMEOUT, and WDIOC_SETOPTIONS ioctls
+ * 09/8 – 2003 [wim@iguana.be] cleanup of trailing spaces
+ * use module_param
+ * made timeout (the emulated heartbeat) a module_param
+ * made the keepalive ping an internal subroutine
+ * added MODULE_AUTHOR and MODULE_DESCRIPTION info
+
+
- * Theory of operation:
- * A Watchdog Timer (WDT) is a hardware circuit that can
- * reset the computer system in case of a software fault.
- * You probably knew that already.
-
- * Usually a userspace daemon will notify the kernel WDT driver
- * via the /proc/watchdog special device file that userspace is
- * still alive, at regular intervals. When such a notification
- * occurs, the driver will usually tell the hardware watchdog
- * that everything is in order, and that the watchdog should wait
- * for yet another little while to reset the system.
- * If userspace fails (RAM error, kernel bug, whatever), the
- * notifications cease to occur, and the hardware watchdog will
- * reset the system (causing a reboot) after the timeout occurs.
-
- * This WDT driver is different from the other Linux WDT
- * drivers in several ways:
+ * This WDT driver is different from the other Linux WDT
+ * drivers in the following ways:
+ *) The driver will ping the watchdog by itself, because this
+ * particular WDT has a very short timeout (one second) and it
+ * would be insane to count on any userspace daemon always
+ * getting scheduled within that time frame.
- *) This driver expects the userspace daemon to send a specific
- * character code ('V') to /dev/watchdog before closing the
- * /dev/watchdog file. If the userspace daemon closes the file
- * without sending this special character, the driver will assume
- * that the daemon (and userspace in general) died, and will
- * stop pinging the WDT without disabling it first. This will
- * cause a reboot.
-
- * Why `V' ? Well, `V' is the character in ASCII for the value 86,
- * and we all know that 86 is _the_ most random number in the universe.
- * Therefore it is the letter that has the slightest chance of occurring
- * by chance, when the system becomes corrupted.
+
+*/
```

Linux-Kernel: [PATCH] 2.6.0-test3 – Watchdog patches (sbc60xxwdt.c and advantechwdt.c)

```
@@ -73,11 +60,12 @@
#include <asm/system.h>

#define OUR_NAME "sbc60xxwdt"
+#define PFX OUR_NAME ": "

/*
 * You must set these – The driver cannot probe for the settings
 */
-
+
#define WDT_STOP 0x45
#define WDT_START 0x443

@@ -92,19 +80,16 @@
/*
 * We must not require too good response from the userspace daemon.
 * Here we require the userspace daemon to send us a heartbeat
- * char to /dev/watchdog every 10 seconds.
- * If the daemon pulses us every 5 seconds, we can still afford
+ * char to /dev/watchdog every 30 seconds.
+ * If the daemon pulses us every 25 seconds, we can still afford
 * a 5 second scheduling delay on the (high priority) daemon. That
 * should be sufficient for a box under any load.
 */

-#define WDT_HEARTBEAT (HZ * 10)
-
-static void wdt_timer_ping(unsigned long);
-static struct timer_list timer;
-static unsigned long next_heartbeat;
-static int wdt_is_open;
-static int wdt_expect_close;
+#define WATCHDOG_TIMEOUT 30 /* 30 sec default timeout */
+static int timeout = WATCHDOG_TIMEOUT; /* in seconds, will be multiplied by HZ to get seconds to
wait for a ping */
+module_param(timeout, int, 0);
+MODULE_PARM_DESC(timeout, "Watchdog timeout in seconds. (1<=timeout<=3600,
default=WATCHDOG_TIMEOUT)");

#ifdef CONFIG_WATCHDOG_NOWAYOUT
static int nowayout = 1;
@@ -115,6 +100,12 @@
module_param(nowayout, int, 0);
MODULE_PARM_DESC(nowayout, "Watchdog cannot be stopped once started
(default=CONFIG_WATCHDOG_NOWAYOUT)");

+static void wdt_timer_ping(unsigned long);
+static struct timer_list timer;
+static unsigned long next_heartbeat;
+static unsigned long wdt_is_open;
```

```

+static char wdt_expect_close;
+
+/*
+ * Whack the dog
+ */
@@ -122,9 +113,9 @@
static void wdt_timer_ping(unsigned long data)
{
    /* If we got a heartbeat pulse within the WDT_US_INTERVAL
- * we agree to ping the WDT
+ * we agree to ping the WDT
    */
- if(time_before(jiffies, next_heartbeat))
+ if(time_before(jiffies, next_heartbeat))
    {
        /* Ping the WDT by reading from WDT_START */
        inb_p(WDT_START);
@@ -132,22 +123,22 @@
        timer.expires = jiffies + WDT_INTERVAL;
        add_timer(&timer);
    } else {
- printk(OUR_NAME ": Heartbeat lost! Will not ping the watchdog\n");
+ printk(KERN_WARNING PFX "Heartbeat lost! Will not ping the watchdog\n");
    }
}

-/*
+/*
+ * Utility routines
+ */

static void wdt_startup(void)
{
- next_heartbeat = jiffies + WDT_HEARTBEAT;
+ next_heartbeat = jiffies + (timeout * HZ);

    /* Start the timer */
- timer.expires = jiffies + WDT_INTERVAL;
+ timer.expires = jiffies + WDT_INTERVAL;
    add_timer(&timer);
- printk(OUR_NAME ": Watchdog timer is now enabled.\n");
+ printk(KERN_INFO PFX "Watchdog timer is now enabled.\n");
}

static void wdt_turnoff(void)
@@ -155,9 +146,14 @@
    /* Stop the timer */
    del_timer(&timer);
    inb_p(WDT_STOP);
- printk(OUR_NAME ": Watchdog timer is now disabled...\n");
+ printk(KERN_INFO PFX "Watchdog timer is now disabled...\n");
}

```

```

}

+static void wdt_keepalive(void)
+{
+ /* user land ping */
+ next_heartbeat = jiffies + (timeout * HZ);
+}

/*
 * /dev/watchdog handling
 @@ -169,63 +165,58 @@
     if(ppos != &file->f_pos)
         return -ESPIPE;

- /* See if we got the magic character */
- if(count)
+ /* See if we got the magic character 'V' and reload the timer */
+ if(count)
     {
- size_t ofs;
-
- /* note: just in case someone wrote the magic character
- * five months ago... */
- wdt_expect_close = 0;
-
- /* now scan */
- for(ofs = 0; ofs != count; ofs++)
+ if (!nowayout)
     {
- char c;
- if(get_user(c, buf+ofs))
- return -EFAULT;
- if(c == 'V')
- wdt_expect_close = 1;
+ size_t ofs;
+
+ /* note: just in case someone wrote the magic character
+ * five months ago... */
+ wdt_expect_close = 0;
+
+ /* scan to see whether or not we got the magic character */
+ for(ofs = 0; ofs != count; ofs++)
+ {
+ char c;
+ if(get_user(c, buf+ofs))
+ return -EFAULT;
+ if(c == 'V')
+ wdt_expect_close = 42;
+ }
     }
+

```

Linux-Kernel: [PATCH] 2.6.0-test3 – Watchdog patches (sbc60xxwdt.c and advantechwdt.c)

```
        /* Well, anyhow someone wrote to us, we should return that favour */
- next_heartbeat = jiffies + WDT_HEARTBEAT;
- return 1;
+ wdt_keepalive();
    }
- return 0;
+ return count;
}

static int fop_open(struct inode * inode, struct file * file)
{
- switch(minor(inode->i_rdev))
- {
- case WATCHDOG_MINOR:
- /* Just in case we're already talking to someone... */
- if(wdt_is_open)
- return -EBUSY;
- if (nowayout)
- __module_get(THIS_MODULE);
- /* Good, fire up the show */
- wdt_is_open = 1;
- wdt_startup();
- return 0;
+ /* Just in case we're already talking to someone... */
+ if(test_and_set_bit(0, &wdt_is_open))
+ return -EBUSY;

- default:
- return -ENODEV;
- }
+ if (nowayout)
+ __module_get(THIS_MODULE);
+
+ /* Good, fire up the show */
+ wdt_startup();
+ return 0;
}

static int fop_close(struct inode * inode, struct file * file)
{
- if(minor(inode->i_rdev) == WATCHDOG_MINOR)
- {
- if(wdt_expect_close && !nowayout)
- wdt_turnoff();
- else {
- del_timer(&timer);
- printk(OUR_NAME ": device file closed unexpectedly. Will not stop the WDT!\n");
- }
+ if(wdt_expect_close == 42)
+ wdt_turnoff();
+ else {
```

```

+ del_timer(&timer);
+ printk(KERN_CRIT PFX "device file closed unexpectedly. Will not stop the WDT!\n");
    }
- wdt_is_open = 0;
+ clear_bit(0, &wdt_is_open);
+ wdt_expect_close = 0;
    return 0;
}

@@ -234,20 +225,58 @@
{
    static struct watchdog_info ident=
    {
- .options = WDIOF_MAGICCLOSE,
+ .options = WDIOF_KEEPAVAILABLE | WDIOF_SETTIMEOUT | WDIOF_MAGICCLOSE,
    .firmware_version = 1,
- .identity = "SB60xx"
+ .identity = "SBC60xx",
    };
-
+
    switch(cmd)
    {
        default:
            return -ENOTTY;
        case WDIOG_GETSUPPORT:
            return copy_to_user((struct watchdog_info *)arg, &ident, sizeof(ident))?-EFAULT:0;
+ case WDIOG_GETSTATUS:
+ case WDIOG_GETBOOTSTATUS:
+ return put_user(0, (int *)arg);
        case WDIOG_KEEPAVAILABLE:
- next_heartbeat = jiffies + WDT_HEARTBEAT;
+ wdt_keepalive();
            return 0;
+ case WDIOG_SETOPTIONS:
+ {
+ int new_options, retval = -EINVAL;
+
+ if(get_user(new_options, (int *)arg))
+ return -EFAULT;
+
+ if(new_options & WDIOG_DISABLECARD) {
+ wdt_turnoff();
+ retval = 0;
+ }
+
+ if(new_options & WDIOG_ENABLECARD) {
+ wdt_startup();
+ retval = 0;
+ }
+

```

```

+ return retval;
+ }
+ case WDIOC_SETTIMEOUT:
+ {
+ int new_timeout;
+
+ if(get_user(new_timeout, (int *)arg))
+ return -EFAULT;
+
+ if(new_timeout < 1 || new_timeout > 3600) /* arbitrary upper limit */
+ return -EINVAL;
+
+ timeout = new_timeout;
+ wdt_keepalive();
+ /* Fall through */
+ }
+ case WDIOC_GETTIMEOUT:
+ return put_user(timeout, (int *)arg);
+ }
+ }

@@ -257,13 +286,13 @@
     .write = fop_write,
     .open = fop_open,
     .release = fop_close,
- .ioctl = fop_ioctl
+ .ioctl = fop_ioctl,
};

static struct miscdevice wdt_miscdev = {
    .minor = WATCHDOG_MINOR,
    .name = "watchdog",
- .fops = &wdt_fops
+ .fops = &wdt_fops,
};

/*
@@ -273,21 +302,21 @@
static int wdt_notify_sys(struct notifier_block *this, unsigned long code,
    void *unused)
{
- if(code==SYS_DOWN || code==SYS_HALT)
+ if(code==SYS_DOWN || code==SYS_HALT)
    wdt_turnoff();
    return NOTIFY_DONE;
}
-
+
/*
* The WDT needs to learn about soft shutdowns in order to
- * turn the timebomb registers off.

```

```

+ * turn the timebomb registers off.
+ */
-
+
static struct notifier_block wdt_notifier=
{
    .notifier_call = wdt_notify_sys,
    .next = NULL,
- .priority = 0
+ .priority = 0,
};

static void __exit sbc60xxwdt_unload(void)
@@ -312,6 +341,13 @@
    if (!request_region(WDT_START, 1, "SBC 60XX WDT"))
        goto err_out_region1;

+ if(timeout < 1 || timeout > 3600) /* arbitrary upper limit */
+ {
+ timeout = WATCHDOG_TIMEOUT;
+ printk (KERN_INFO PFX "timeout value must be 1<=x<=3600, using %d\n",
+ timeout);
+ }
+
    init_timer(&timer);
    timer.function = wdt_timer_ping;
    timer.data = 0;
@@ -324,8 +360,8 @@
    if (rc)
        goto err_out_miscdev;

- printk(KERN_INFO OUR_NAME ": WDT driver for 60XX single board computer initialised.\n");
-
+ printk(KERN_INFO PFX "WDT driver for 60XX single board computer initialised.\n");
+
    return 0;

err_out_miscdev:
@@ -341,4 +377,6 @@
module_init(sbc60xxwdt_init);
module_exit(sbc60xxwdt_unload);

+MODULE_AUTHOR("Jakob Oestergaard <jakob@unthought.net>");
+MODULE_DESCRIPTION("60xx Single Board Computer Watchdog Timer driver");
MODULE_LICENSE("GPL");
diff -Nru a/drivers/char/watchdog/sbc60xxwdt.c b/drivers/char/watchdog/sbc60xxwdt.c
--- a/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:40:42 2003
+++ b/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:40:42 2003
@@ -89,7 +89,7 @@
#define WATCHDOG_TIMEOUT 30 /* 30 sec default timeout */
static int timeout = WATCHDOG_TIMEOUT; /* in seconds, will be multiplied by HZ to get seconds to wait

```

```

for a ping */
module_param(timeout, int, 0);
-MODULE_PARM_DESC(timeout, "Watchdog timeout in seconds. (1<=timeout<=3600,
default=WATCHDOG_TIMEOUT)");
+MODULE_PARM_DESC(timeout, "Watchdog timeout in seconds. (1<=timeout<=3600, default="
__MODULE_STRING(WATCHDOG_TIMEOUT) " "));

#ifdef CONFIG_WATCHDOG_NOWAYOUT
static int nowayout = 1;
diff -Nru a/drivers/char/watchdog/sbc60xxwdt.c b/drivers/char/watchdog/sbc60xxwdt.c
--- a/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:41:01 2003
+++ b/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:41:01 2003
@@ -30,6 +30,7 @@
 * use module_param
 * made timeout (the emulated heartbeat) a module_param
 * made the keepalive ping an internal subroutine
+ * made wdt_stop and wdt_start module params
 * added MODULE_AUTHOR and MODULE_DESCRIPTION info
 *
 *
@@ -66,8 +67,13 @@
 * You must set these – The driver cannot probe for the settings
 */

-#define WDT_STOP 0x45
-#define WDT_START 0x443
+static int wdt_stop = 0x45;
+module_param(wdt_stop, int, 0);
+MODULE_PARM_DESC(wdt_stop, "SBC60xx WDT 'stop' io port (default 0x45)");
+
+static int wdt_start = 0x443;
+module_param(wdt_start, int, 0);
+MODULE_PARM_DESC(wdt_start, "SBC60xx WDT 'start' io port (default 0x443)");

/*
 * The 60xx board can use watchdog timeout values from one second
@@ -117,8 +123,8 @@
 */
if(time_before(jiffies, next_heartbeat))
{
- /* Ping the WDT by reading from WDT_START */
- inb_p(WDT_START);
+ /* Ping the WDT by reading from wdt_start */
+ inb_p(wdt_start);
/* Re-set the timer interval */
timer.expires = jiffies + WDT_INTERVAL;
add_timer(&timer);
@@ -145,7 +151,7 @@
{
/* Stop the timer */
del_timer(&timer);

```

Linux-Kernel: [PATCH] 2.6.0-test3 – Watchdog patches (sbc60xxwdt.c and advantechwdt.c)

```
– inb_p(WDT_STOP);
+ inb_p(wdt_stop);
  printk(KERN_INFO PFX "Watchdog timer is now disabled...\n");
}
```

```
@@ –327,19 +333,24 @@
  misc_deregister(&wdt_miscdev);
```

```
  unregister_reboot_notifier(&wdt_notifier);
– release_region(WDT_START,1);
–// release_region(WDT_STOP,1);
+ if ((wdt_stop != 0x45) && (wdt_stop != wdt_start))
+ release_region(wdt_stop,1);
+ release_region(wdt_start,1);
}
```

```
static int __init sbc60xxwdt_init(void)
{
```

```
  int rc = –EBUSY;
```

```
–// We cannot reserve 0x45 – the kernel already has!
–// if (!request_region(WDT_STOP, 1, "SBC 60XX WDT"))
–// goto err_out;
– if (!request_region(WDT_START, 1, "SBC 60XX WDT"))
– goto err_out_region1;
+ if (!request_region(wdt_start, 1, "SBC 60XX WDT"))
+ goto err_out;
+
+ /* We cannot reserve 0x45 – the kernel already has! */
+ if ((wdt_stop != 0x45) && (wdt_stop != wdt_start))
+ {
+ if (!request_region(wdt_stop, 1, "SBC 60XX WDT"))
+ goto err_out_region1;
+ }
```

```
  if(timeout < 1 || timeout > 3600) /* arbitrary upper limit */
  {
```

```
@@ –367,10 +378,11 @@
```

```
err_out_miscdev:
  misc_deregister(&wdt_miscdev);
err_out_region2:
– release_region(WDT_START,1);
+ if ((wdt_stop != 0x45) && (wdt_stop != wdt_start))
+ release_region(wdt_stop,1);
err_out_region1:
– release_region(WDT_STOP,1);
–/* err_out: */
+ release_region(wdt_start,1);
+err_out:
  return rc;
}
```

```

diff -Nru a/drivers/char/watchdog/sbc60xxwdt.c b/drivers/char/watchdog/sbc60xxwdt.c
--- a/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:41:20 2003
+++ b/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:41:20 2003
@@ -343,19 +343,28 @@
     int rc = -EBUSY;

     if (!request_region(wdt_start, 1, "SBC 60XX WDT"))
+ {
+ printk(KERN_ERR PFX "I/O address 0x%04x already in use\n",
+ wdt_start);
+ rc = -EIO;
+     goto err_out;
+ }

     /* We cannot reserve 0x45 – the kernel already has! */
     if ((wdt_stop != 0x45) && (wdt_stop != wdt_start))
     {
         if (!request_region(wdt_stop, 1, "SBC 60XX WDT"))
+ {
+ printk(KERN_ERR PFX "I/O address 0x%04x already in use\n",
+ wdt_stop);
+     goto err_out_region1;
+ }
     }

     if(timeout < 1 || timeout > 3600) /* arbitrary upper limit */
     {
         timeout = WATCHDOG_TIMEOUT;
- printk (KERN_INFO PFX "timeout value must be 1<=x<=3600, using %d\n",
+ printk(KERN_INFO PFX "timeout value must be 1<=x<=3600, using %d\n",
             timeout);
     }

@@ -365,11 +374,19 @@

     rc = misc_register(&wdt_miscdev);
     if (rc)
+ {
+ printk(KERN_ERR PFX "cannot register miscdev on minor=%d (err=%d)\n",
+ wdt_miscdev.minor, rc);
+     goto err_out_region2;
+ }

     rc = register_reboot_notifier(&wdt_notifier);
     if (rc)
+ {
+ printk(KERN_ERR PFX "cannot register reboot notifier (err=%d)\n",
+ rc);
+     goto err_out_miscdev;
+ }

```

```
printk(KERN_INFO PFX "WDT driver for 60XX single board computer initialised.\n");
```

```
diff -Nru a/drivers/char/watchdog/sbc60xxwdt.c b/drivers/char/watchdog/sbc60xxwdt.c
```

```
--- a/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:41:39 2003
```

```
+++ b/drivers/char/watchdog/sbc60xxwdt.c Sat Aug 9 16:41:39 2003
```

```
@@ -342,6 +342,13 @@
```

```
{
```

```
int rc = -EBUSY;
```

```
+ if(timeout < 1 || timeout > 3600) /* arbitrary upper limit */
```

```
+ {
```

```
+ timeout = WATCHDOG_TIMEOUT;
```

```
+ printk(KERN_INFO PFX "timeout value must be 1<=x<=3600, using %d\n",
```

```
+ timeout);
```

```
+ }
```

```
+
```

```
if (!request_region(wdt_start, 1, "SBC 60XX WDT"))
```

```
{
```

```
printk(KERN_ERR PFX "I/O address 0x%04x already in use\n",
```

```
@@ -361,13 +368,6 @@
```

```
}
```

```
}
```

```
- if(timeout < 1 || timeout > 3600) /* arbitrary upper limit */
```

```
- {
```

```
- timeout = WATCHDOG_TIMEOUT;
```

```
- printk(KERN_INFO PFX "timeout value must be 1<=x<=3600, using %d\n",
```

```
- timeout);
```

```
- }
```

```
-
```

```
init_timer(&timer);
```

```
timer.function = wdt_timer_ping;
```

```
timer.data = 0;
```

```
@@ -388,7 +388,8 @@
```

```
goto err_out_miscdev;
```

```
}
```

```
- printk(KERN_INFO PFX "WDT driver for 60XX single board computer initialised.\n");
```

```
+ printk(KERN_INFO PFX "WDT driver for 60XX single board computer initialised. timeout=%d sec  
(nowayout=%d)\n",
```

```
+ timeout, nowayout);
```

```
return 0;
```

```
diff -Nru a/drivers/char/watchdog/advantechwdt.c b/drivers/char/watchdog/advantechwdt.c
```

```
--- a/drivers/char/watchdog/advantechwdt.c Sat Aug 9 16:41:58 2003
```

```
+++ b/drivers/char/watchdog/advantechwdt.c Sat Aug 9 16:41:58 2003
```

```
@@ -44,6 +44,7 @@
```

```
#include <asm/system.h>
```

Linux-Kernel: [PATCH] 2.6.0-test3 – Watchdog patches (sbc60xxwdt.c and advantechwdt.c)

```

#define WATCHDOG_NAME "Advantech WDT"
+#define PFX WATCHDOG_NAME ": "
#define WATCHDOG_TIMEOUT 60 /* 60 sec default timeout */

static unsigned long advwdt_is_open;
@@ -70,7 +71,7 @@

static int timeout = WATCHDOG_TIMEOUT; /* in seconds */
module_param(timeout, int, 0);
-MODULE_PARM_DESC(timeout, "Watchdog timeout in seconds. 1<= timeout <=63, default=60.");
+MODULE_PARM_DESC(timeout, "Watchdog timeout in seconds. 1<= timeout <=63, default="
__MODULE_STRING(WATCHDOG_TIMEOUT) ".");

#ifdef CONFIG_WATCHDOG_NOWAYOUT
static int nowayout = 1;
@@ -206,7 +207,7 @@
    if (adv_expect_close == 42) {
        advwdt_disable();
    } else {
- printk(KERN_CRIT WATCHDOG_NAME ": Unexpected close, not stopping watchdog!\n");
+ printk(KERN_CRIT PFX "Unexpected close, not stopping watchdog!\n");
        advwdt_ping();
    }
    clear_bit(0, &advwdt_is_open);
@@ -268,13 +269,13 @@

    if (timeout < 1 || timeout > 63) {
        timeout = WATCHDOG_TIMEOUT;
- printk (KERN_INFO WATCHDOG_NAME ": timeout value must be 1<=x<=63, using %d\n",
+ printk (KERN_INFO PFX "timeout value must be 1<=x<=63, using %d\n",
        timeout);
    }

    if (wdt_stop != wdt_start) {
        if (!request_region(wdt_stop, 1, WATCHDOG_NAME)) {
- printk (KERN_ERR WATCHDOG_NAME ": I/O address 0x%04x already in use\n",
+ printk (KERN_ERR PFX "I/O address 0x%04x already in use\n",
        wdt_stop);
        ret = -EIO;
        goto out;
@@ -282,7 +283,7 @@
    }

    if (!request_region(wdt_start, 1, WATCHDOG_NAME)) {
- printk (KERN_ERR WATCHDOG_NAME ": I/O address 0x%04x already in use\n",
+ printk (KERN_ERR PFX "I/O address 0x%04x already in use\n",
        wdt_start);
        ret = -EIO;
        goto unreg_stop;
@@ -290,19 +291,19 @@

```

Linux-Kernel: [PATCH] 2.6.0-test3 – Watchdog patches (sbc60xxwdt.c and advantechwdt.c)

```
ret = register_reboot_notifier(&advwdt_notifier);
if (ret != 0) {
- printk (KERN_ERR WATCHDOG_NAME ": cannot register reboot notifier (err=%d)\n",
+ printk (KERN_ERR PFX "cannot register reboot notifier (err=%d)\n",
        ret);
    goto unreg_regions;
}

ret = misc_register(&advwdt_miscdev);
if (ret != 0) {
- printk (KERN_ERR WATCHDOG_NAME ": cannot register miscdev on minor=%d (err=%d)\n",
+ printk (KERN_ERR PFX "cannot register miscdev on minor=%d (err=%d)\n",
        WATCHDOG_MINOR, ret);
    goto unreg_reboot;
}

- printk (KERN_INFO WATCHDOG_NAME ": initialized. timeout=%d sec (nowayout=%d)\n",
+ printk (KERN_INFO PFX "initialized. timeout=%d sec (nowayout=%d)\n",
        timeout, nowayout);
```

out:

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>