

## Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-01/msg09329.html>

---

- *From:* Thomas Renninger <trenn@xxxxxxx>
  - *Date:* Wed, 23 Jan 2008 18:38:37 +0100
- 

On Sun, 2008-01-20 at 02:23 +0200, Pekka Enberg wrote:

Hi Thomas,

...

When krealloc() returns NULL, there wasn't enough memory to fit the new size but the original memory region remains unchanged.

...

Thanks Pekka, this one should be better now:

The patch is against latest 2.6.24-rc8 not -mm tree:

Allocate pnp resources dynamically via krealloc

Latest BIOS ACPI PNP device resource descriptions may have (especially on the general device PNP0c02) more than 20 IO port resources.

Reserve the space in a static array wastes a lot of memory on every PNP device and is not a real option as the number of IO ports could be much greater than 20.

This approach allocates the memory for PNP resources at runtime.

The memory is reallocated in e.g. 8 (for IO port) resource portions.

That means that the previously allocated pointers will get a new address.

Therefore this address must not be stored and/or used as long as krealloc might still allocate new resources.

From what I have seen, there is a patch in -mm that gets rid of

registering resources automatically and pass the pointers from pnp\_resource\_table to request\_region.

While this should still work (only disabled devices where the regions should have been unregistered should be modifiable) it is potential dangerous: once you realloc pnp\_resource\_table pointers you end up with invalid pointers in the kernel/resource.c implemented list.

Finding this could get difficult as you get really ugly phenomenons with

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

corrupted memory...

The patch also needs another patch from Rene Herman:

"[PATCH] sound/isa: kill pnp\_resource\_change."

Sent to the alsa-devel list and it's already included in Takashi's tree for 2.6.25 inclusion.

The function pnp\_resource\_change is a nop now and immediately returns to avoid compile errors. It can be removed as soon as everything is merged together.

This has been tested on rather new machines on i386 and x86\_64.

The first with pnpbios also compiled in and forced a test with pnpacpi=off.

isapnp is totally untested. Also the sysfs is rather untested.

I tried to play a bit with the sysfs interface and wanted override resources without much success:

Unloaded parport\_pc driver, then the corresponding pnp device changed state from active to disabled and now I should have been able to modify BIOS settings, but it did not work. I expect this is rather unused and could have been broken before and I did not want to loose more time with it.

Maybe someone else knows more here.

Signed-off-by: Thomas Renninger <trenn@xxxxxxx>

```

---
drivers/pnp/core.c | 2
drivers/pnp/interface.c | 44 +++-
drivers/pnp/isapnp/core.c | 33 +-
drivers/pnp/manager.c | 402 ++++++-----
drivers/pnp/pnpacpi/rsparser.c | 151 ++++++-----
drivers/pnp/pnpbios/rsparser.c | 112 ++++++-----
drivers/pnp/resource.c | 16 -
drivers/pnp/support.c | 11 -
drivers/pnp/system.c | 4
include/linux/pnp.h | 51 +++--
10 files changed, 587 insertions(+), 239 deletions(-)

```

Index: linux-2.6.23/include/linux/pnp.h

```

=====
--- linux-2.6.23.orig/include/linux/pnp.h
+++ linux-2.6.23/include/linux/pnp.h
@@ -13,10 +13,6 @@
#include <linux/errno.h>
#include <linux/mod_devicetable.h>

-#define PNP_MAX_PORT 40
-#define PNP_MAX_MEM 12
-#define PNP_MAX_IRQ 2
-#define PNP_MAX_DMA 2

```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
#define PNP_NAME_LEN 50

struct pnp_protocol;
@@ -26,13 +22,24 @@ struct pnp_dev;
* Resource Management
*/

-/* Use these instead of directly reading pnp_dev to get resource information */
+/*
+ * NULL pointer alarm: always check with pnp_port_ok or pnp_port_valid before
+ * accessing start/end/flags/len values or you might access not allocated mem.
+ * Same for mem, irq and dma macros
+ *
+ * Pointers are not static and they might change, do not store addresses
+ * of resources in the pnp resource table!
+ */
+
+#define pnp_port_ok(dev,bar) ((dev)->res.allocated_ports > (bar))
+
#define pnp_port_start(dev,bar) ((dev)->res.port_resource[(bar)].start)
#define pnp_port_end(dev,bar) ((dev)->res.port_resource[(bar)].end)
#define pnp_port_flags(dev,bar) ((dev)->res.port_resource[(bar)].flags)
#define pnp_port_valid(dev,bar) \
+ (pnp_port_ok((dev),(bar)) && \
((pnp_port_flags((dev),(bar)) & (IORESOURCE_IO | IORESOURCE_UNSET)) \
- == IORESOURCE_IO)
+ == IORESOURCE_IO))
#define pnp_port_len(dev,bar) \
((pnp_port_start((dev),(bar)) == 0 && \
pnp_port_end((dev),(bar)) == \
@@ -41,12 +48,14 @@ struct pnp_dev;
(pnp_port_end((dev),(bar)) - \
pnp_port_start((dev),(bar)) + 1))

+#define pnp_mem_ok(dev,bar) ((dev)->res.allocated_mems > (bar))
#define pnp_mem_start(dev,bar) ((dev)->res.mem_resource[(bar)].start)
#define pnp_mem_end(dev,bar) ((dev)->res.mem_resource[(bar)].end)
#define pnp_mem_flags(dev,bar) ((dev)->res.mem_resource[(bar)].flags)
#define pnp_mem_valid(dev,bar) \
+ (pnp_mem_ok((dev),(bar)) && \
((pnp_mem_flags((dev),(bar)) & (IORESOURCE_MEM | IORESOURCE_UNSET)) \
- == IORESOURCE_MEM)
+ == IORESOURCE_MEM))
#define pnp_mem_len(dev,bar) \
((pnp_mem_start((dev),(bar)) == 0 && \
pnp_mem_end((dev),(bar)) == \
@@ -55,17 +64,21 @@ struct pnp_dev;
(pnp_mem_end((dev),(bar)) - \
pnp_mem_start((dev),(bar)) + 1))

+#define pnp_irq_ok(dev,bar) ((dev)->res.allocated_irqs > (bar))
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
#define pnp_irq(dev,bar) ((dev)->res.irq_resource[(bar)].start)
#define pnp_irq_flags(dev,bar) ((dev)->res.irq_resource[(bar)].flags)
#define pnp_irq_valid(dev,bar) \
+ (pnp_irq_ok((dev),(bar)) && \
((pnp_irq_flags((dev),(bar)) & (IORESOURCE_IRQ | IORESOURCE_UNSET)) \
- == IORESOURCE_IRQ) \
+ == IORESOURCE_IRQ))

+#define pnp_dma_ok(dev,bar) ((dev)->res.allocated_dmas > (bar))
#define pnp_dma(dev,bar) ((dev)->res.dma_resource[(bar)].start)
#define pnp_dma_flags(dev,bar) ((dev)->res.dma_resource[(bar)].flags)
#define pnp_dma_valid(dev,bar) \
+ (pnp_dma_ok((dev),(bar)) && \
((pnp_dma_flags((dev),(bar)) & (IORESOURCE_DMA | IORESOURCE_UNSET)) \
- == IORESOURCE_DMA) \
+ == IORESOURCE_DMA))

#define PNP_PORT_FLAG_16BITADDR (1<<0)
#define PNP_PORT_FLAG_FIXED (1<<1)
@@ -119,10 +132,14 @@ struct pnp_option {
};

struct pnp_resource_table {
- struct resource port_resource[PNP_MAX_PORT];
- struct resource mem_resource[PNP_MAX_MEM];
- struct resource dma_resource[PNP_MAX_DMA];
- struct resource irq_resource[PNP_MAX_IRQ];
+ struct resource *port_resource;
+ unsigned int allocated_ports;
+ struct resource *mem_resource;
+ unsigned int allocated_mems;
+ struct resource *dma_resource;
+ unsigned int allocated_dmas;
+ struct resource *irq_resource;
+ unsigned int allocated_irqs;
};

/*
@@ -364,6 +381,7 @@ int pnp_device_attach(struct pnp_dev *pn
void pnp_device_detach(struct pnp_dev *pnp_dev);
extern struct list_head pnp_global;
extern int pnp_platform_devices;
+extern int pnp_bios_data_parsed;

/* multidevice card support */
int pnp_add_card(struct pnp_card *card);
@@ -398,6 +416,8 @@ int pnp_activate_dev(struct pnp_dev *dev
int pnp_disable_dev(struct pnp_dev *dev);
void pnp_resource_change(struct resource *resource, resource_size_t start,
resource_size_t size);
+int pnp_assign_resource(struct pnp_resource_table *table, struct resource *res);
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

+

```
/* protocol helpers */
int pnp_is_active(struct pnp_dev *dev);
@@ -445,6 +465,7 @@ static inline int pnp_stop_dev(struct pn
static inline int pnp_activate_dev(struct pnp_dev *dev) { return -ENODEV; }
static inline int pnp_disable_dev(struct pnp_dev *dev) { return -ENODEV; }
static inline void pnp_resource_change(struct resource *resource, resource_size_t start, resource_size_t size) {
}
+static inline int pnp_assign_resource(struct pnp_resource_table *table, struct resource *res) { }
```

```
/* protocol helpers */
static inline int pnp_is_active(struct pnp_dev *dev) { return 0; }
@@ -460,9 +481,11 @@ static inline void pnp_unregister_driver
#define pnp_warn(format, arg...) printk(KERN_WARNING "pnp: " format "\n" , ## arg)
```

```
#ifdef CONFIG_PNP_DEBUG
-#define pnp_dbg(format, arg...) printk(KERN_DEBUG "pnp: " format "\n" , ## arg)
+#define pnp_dbg(format, arg...) printk(KERN_INFO "pnp: " format "\n" , ## arg)
+void pnp_dump_resources(struct pnp_dev *dev);
#else
#define pnp_dbg(format, arg...) do { } while (0)
+static inline void pnp_dump_resources(struct pnp_dev *dev) { }
#endif
```

```
#endif /* __KERNEL__ */
```

Index: linux-2.6.23/drivers/pnp/interface.c

=====

--- linux-2.6.23.orig/drivers/pnp/interface.c

+++ linux-2.6.23/drivers/pnp/interface.c

```
@@ -264,7 +264,7 @@ static ssize_t pnp_show_current_resource
else
pnp_printf(buffer, "disabled\n");
```

```
- for (i = 0; i < PNP_MAX_PORT; i++) {
+ for (i = 0; pnp_port_ok(dev, i); i++) {
if (pnp_port_valid(dev, i)) {
pnp_printf(buffer, "io");
if (pnp_port_flags(dev, i) & IORESOURCE_DISABLED)
@@ -277,7 +277,7 @@ static ssize_t pnp_show_current_resource
i));
}
}
- for (i = 0; i < PNP_MAX_MEM; i++) {
+ for (i = 0; pnp_mem_ok(dev, i); i++) {
if (pnp_mem_valid(dev, i)) {
pnp_printf(buffer, "mem");
if (pnp_mem_flags(dev, i) & IORESOURCE_DISABLED)
@@ -290,7 +290,7 @@ static ssize_t pnp_show_current_resource
i));
}
}
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
}
- for (i = 0; i < PNP_MAX_IRQ; i++) {
+ for (i = 0; pnp_irq_ok(dev, i); i++) {
if (pnp_irq_valid(dev, i)) {
pnp_printf(buffer, "irq");
if (pnp_irq_flags(dev, i) & IORESOURCE_DISABLED)
@@ -300,7 +300,7 @@ static ssize_t pnp_show_current_resource
(unsigned long long)pnp_irq(dev, i));
}
}
- for (i = 0; i < PNP_MAX_DMA; i++) {
+ for (i = 0; pnp_dma_ok(dev, i); i++) {
if (pnp_dma_valid(dev, i)) {
pnp_printf(buffer, "dma");
if (pnp_dma_flags(dev, i) & IORESOURCE_DISABLED)
@@ -381,6 +381,13 @@ pnp_set_current_resources(struct device
buf += 2;
while (isspace(*buf))
++buf;
+ if (!pnp_port_ok(dev, nport)) {
+ buf++;
+ pnp_err("Cannot manually set port "
+ "resource %d for device %s",
+ nport, dev->name);
+ continue;
+ }
dev->res.port_resource[nport].start =
simple_strtoul(buf, &buf, 0);
while (isspace(*buf))
@@ -397,14 +404,19 @@ pnp_set_current_resources(struct device
dev->res.port_resource[nport].flags =
IORESOURCE_IO;
nport++;
- if (nport >= PNP_MAX_PORT)
- break;
continue;
}
if (!strnicmp(buf, "mem", 3)) {
buf += 3;
while (isspace(*buf))
++buf;
+ if (!pnp_mem_ok(dev, nmem)) {
+ buf++;
+ pnp_err("Cannot manually set mem "
+ "resource %d for device %s",
+ nmem, dev->name);
+ continue;
+ }
dev->res.mem_resource[nmem].start =
simple_strtoul(buf, &buf, 0);
while (isspace(*buf))
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
@@ -421,36 +433,44 @@ pnp_set_current_resources(struct device
dev->res.mem_resource[nmem].flags =
IORESOURCE_MEM;
nmem++;
- if (nmem >= PNP_MAX_MEM)
- break;
continue;
}
if (!strnicmp(buf, "irq", 3)) {
buf += 3;
while (isspace(*buf))
++buf;
+ if (!pnp_irq_ok(dev, nirq)) {
+ buf++;
+ pnp_err("Cannot manually set irq "
+ "resource %d for device %s",
+ nirq, dev->name);
+ continue;
+ }
dev->res.irq_resource[nirq].start =
dev->res.irq_resource[nirq].end =
simple_strtoul(buf, &buf, 0);
dev->res.irq_resource[nirq].flags =
IORESOURCE_IRQ;
nirq++;
- if (nirq >= PNP_MAX_IRQ)
- break;
continue;
}
if (!strnicmp(buf, "dma", 3)) {
buf += 3;
while (isspace(*buf))
++buf;
+ if (!pnp_dma_ok(dev, ndma)) {
+ buf++;
+ pnp_err("Cannot manually set dma "
+ "resource %d for device %s",
+ ndma, dev->name);
+ continue;
+ }
dev->res.dma_resource[ndma].start =
dev->res.dma_resource[ndma].end =
simple_strtoul(buf, &buf, 0);
dev->res.dma_resource[ndma].flags =
IORESOURCE_DMA;
ndma++;
- if (ndma >= PNP_MAX_DMA)
- break;
continue;
}
break;
```

## Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

Index: linux-2.6.23/drivers/pnp/manager.c

```
=====
--- linux-2.6.23.orig/drivers/pnp/manager.c
+++ linux-2.6.23/drivers/pnp/manager.c
@@ -14,15 +14,338 @@
#include <linux/bitmap.h>
#include "base.h"

+/* Defines the amount of struct resources that will get (re-)allocated
+ * if the resource table runs out of allocated ports/irqs/dma/mems
+ */
+#define PNP_ALLOC_PORT 8
+#define PNP_ALLOC_MEM 4
+#define PNP_ALLOC_IRQ 2
+#define PNP_ALLOC_DMA 2
+
+DECLARE_MUTEX(pnp_res_mutex);

+#ifdef CONFIG_PNP_DEBUG
+void pnp_dump_resources(struct pnp_dev *dev)
+{
+ int i;
+ pnp_dbg("Resource table dump:");
+ pnp_dbg("Allocated: ports: %d [%p - %p]",
+ dev->res.allocated_ports, dev->res.port_resource,
+ dev->res.port_resource + (dev->res.allocated_ports *
+ sizeof(struct resource)));
+ for (i = 0; pnp_port_ok(dev, i); i++) {
+ pnp_dbg("Port %d: start: 0x%lx - end: 0x%lx - flags: %lu", i,
+ (unsigned long)pnp_port_start(dev, i),
+ (unsigned long)pnp_port_end(dev, i),
+ pnp_port_flags(dev, i));
+ }
+ pnp_dbg("Allocated: mems: %d [%p - %p]",
+ dev->res.allocated_mems, dev->res.mem_resource,
+ dev->res.mem_resource + (dev->res.allocated_mems *
+ sizeof(struct resource)));
+ for (i = 0; pnp_mem_ok(dev, i); i++) {
+ pnp_dbg("Mem %d: start: 0x%lx - end: 0x%lx - flags: %lu", i,
+ (unsigned long)pnp_mem_start(dev, i),
+ (unsigned long)pnp_mem_end(dev, i),
+ pnp_mem_flags(dev, i));
+ }
+ pnp_dbg("Allocated: irqs: %d [%p - %p]",
+ dev->res.allocated_irqs, dev->res.irq_resource,
+ dev->res.irq_resource + (dev->res.allocated_irqs *
+ sizeof(struct resource)));
+ for (i = 0; pnp_irq_ok(dev, i); i++) {
+ pnp_dbg("Irq %d: start: 0x%lx - flags: %lu", i,
+ (unsigned long)pnp_irq(dev, i),
+ pnp_irq_flags(dev, i));
+ }
+ }
+#endif
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version



Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
+ * allocated struct resources in the table.
+ * This would invalidate the addresses passed to request/insert_resource
+ */
+
+static int pnp_alloc_port(struct pnp_resource_table *res)
+{
+ int i;
+ void *ret;
+
+ ret = krealloc(res->port_resource,
+ (sizeof(struct resource) * res->allocated_ports)
+ + (sizeof(struct resource) * PNP_ALLOC_PORT), GFP_KERNEL);
+
+ if (!ret)
+ return -ENOMEM;
+
+ res->port_resource = ret;
+
+ res->allocated_ports += PNP_ALLOC_PORT;
+ for (i = res->allocated_ports - PNP_ALLOC_PORT;
+ i < res->allocated_ports; i++)
+ pnp_init_io(&res->port_resource[i]);
+
+ pnp_dbg("Port allocate: %p - %p; Allocated: %lu bytes, size of"
+ "struct: %lu - allocated ports: %d",
+ res->port_resource,
+ res->port_resource
+ + (sizeof(struct resource) * res->allocated_ports)
+ + (sizeof(struct resource) * PNP_ALLOC_PORT),
+ (unsigned long) (sizeof(struct resource) * res->allocated_ports)
+ + (sizeof(struct resource) * PNP_ALLOC_PORT),
+ (unsigned long) sizeof(struct resource),
+ res->allocated_ports);
+
+ return 0;
+}
+
+static int pnp_alloc_mem(struct pnp_resource_table *res)
+{
+ int i;
+ void *ret;
+
+ ret = krealloc(res->mem_resource,
+ (sizeof(struct resource) * res->allocated_mems)
+ + (sizeof(struct resource) * PNP_ALLOC_MEM), GFP_KERNEL);
+
+ if (!ret)
+ return -ENOMEM;
+
+ res->mem_resource = ret;
+
+}
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
+ res->allocated_mems += PNP_ALLOC_MEM;
+
+ for (i = res->allocated_mems - PNP_ALLOC_MEM; i < res->allocated_mems;
+ i++)
+ pnp_init_mem(&res->mem_resource[i]);
+
+ pnp_dbg("Mem allocate: %p - %p; Allocated: %lu bytes, size of"
+ "struct: %lu - allocated mems: %d",
+ res->mem_resource,
+ res->mem_resource
+ + (sizeof(struct resource) * res->allocated_mems)
+ + (sizeof(struct resource) * PNP_ALLOC_MEM),
+ (unsigned long) (sizeof(struct resource) * res->allocated_mems)
+ + (sizeof(struct resource) * PNP_ALLOC_MEM),
+ (unsigned long) sizeof(struct resource),
+ res->allocated_mems);
+
+ return 0;
+}
+
+static int pnp_alloc_irq(struct pnp_resource_table *res)
+{
+ int i;
+ void *ret;
+
+ ret = krealloc(res->irq_resource,
+ (sizeof(struct resource) * res->allocated_irqs)
+ + (sizeof(struct resource) * PNP_ALLOC_IRQ), GFP_KERNEL);
+
+ if (!ret)
+ return -ENOMEM;
+
+ res->irq_resource = ret;
+
+ res->allocated_irqs += PNP_ALLOC_IRQ;
+ for (i = res->allocated_irqs - PNP_ALLOC_IRQ; i < res->allocated_irqs;
+ i++)
+ pnp_init_irq(&res->irq_resource[i]);
+
+ pnp_dbg("Irq allocate: %p - %p; Allocated: %lu bytes, size of"
+ "struct: %lu - allocated irq: %d",
+ res->irq_resource,
+ res->irq_resource
+ + (sizeof(struct resource) * res->allocated_irqs)
+ + (sizeof(struct resource) * PNP_ALLOC_IRQ),
+ (unsigned long) (sizeof(struct resource) * res->allocated_irqs)
+ + (sizeof(struct resource) * PNP_ALLOC_IRQ),
+ (unsigned long) sizeof(struct resource),
+ res->allocated_irqs);
+ return 0;
+}
```



Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
+ if (res->flags & IORESOURCE_IO) {
+ /* find the next unused table entry */
+ while (i < table->allocated_ports) {
+ if (!(table->port_resource[i].flags
+ & IORESOURCE_UNSET))
+ i++;
+ else
+ break;
+ }
+ /* No unused table entry anymore, allocate new ones */
+ if (table->allocated_ports <= i) {
+ ret = pnp_alloc_port(table);
+ if (ret) {
+ pnp_print_alloc_err("Port", ret, i);
+ return ret;
+ }
+ }
+ memcpy(&table->port_resource[i], res, sizeof(struct resource));
+ } else if (res->flags & IORESOURCE_MEM) {
+ while (i < table->allocated_mems) {
+ if (!(table->mem_resource[i].flags & IORESOURCE_UNSET))
+ i++;
+ else
+ break;
+ }
+
+ if (table->allocated_mems <= i) {
+ ret = pnp_alloc_mem(table);
+ if (ret) {
+ pnp_print_alloc_err("System Memory", ret, i);
+ return ret;
+ }
+ }
+ memcpy(&table->mem_resource[i], res, sizeof(struct resource));
+ } else if (res->flags & IORESOURCE_IRQ) {
+ while (i < table->allocated_irqs) {
+ if (!(table->irq_resource[i].flags & IORESOURCE_UNSET))
+ i++;
+ else
+ break;
+ }
+
+ if (table->allocated_irqs <= i) {
+ ret = pnp_alloc_irq(table);
+ if (ret) {
+ pnp_print_alloc_err("Irq", ret, i);
+ return ret;
+ }
+ }
+ memcpy(&table->irq_resource[i], res, sizeof(struct resource));
+ } else if (res->flags & IORESOURCE_DMA) {
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
+ while (i < table->allocated_dmas) {
+ if (!(table->dma_resource[i].flags & IORESOURCE_UNSET))
+ i++;
+ else
+ break;
+ }
+
+ if (table->allocated_dmas <= i) {
+ ret = pnp_alloc_dma(table);
+ if (ret) {
+ pnp_print_alloc_err("DMA", ret, i);
+ return ret;
+ }
+ }
+ memcpy(&table->dma_resource[i], res, sizeof(struct resource));
+ } else
+ return -EINVAL;
+
+ return 0;
+}
+
+#define pnp_print_assign_err(type, val) \
+ pnp_dbg("%s resource %d not allocated, cannot assign value", \
+ type, val);
+
+
+static int pnp_assign_port(struct pnp_dev *dev, struct pnp_port *rule, int idx)
+{
+ resource_size_t *start, *end;
+ unsigned long *flags;
+
+ - if (idx >= PNP_MAX_PORT) {
+ - dev_err(&dev->dev, "too many I/O port resources\n");
+ + if (!pnp_port_ok(dev, idx)) {
+ + pnp_print_assign_err("Port", idx);
+ /* pretend we were successful so at least the manager won't try again */
+ return 1;
+ }
+ @@ -62,9 +385,8 @@ static int pnp_assign_mem(struct pnp_dev
+ resource_size_t *start, *end;
+ unsigned long *flags;
+
+ - if (idx >= PNP_MAX_MEM) {
+ - dev_err(&dev->dev, "too many memory resources\n");
+ - /* pretend we were successful so at least the manager won't try again */
+ + if (!pnp_mem_ok(dev, idx)) {
+ + pnp_print_assign_err("System Memory", idx);
+ return 1;
+ }
+ }
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
@@ -119,9 +441,8 @@ static int pnp_assign_irq(struct pnp_dev
5, 10, 11, 12, 9, 14, 15, 7, 3, 4, 13, 0, 1, 6, 8, 2
};

- if (idx >= PNP_MAX_IRQ) {
- dev_err(&dev->dev, "too many IRQ resources\n");
- /* pretend we were successful so at least the manager won't try again */
+ if (!pnp_irq_ok(dev, idx)) {
+ pnp_print_assign_err("Irq", idx);
return 1;
}

@@ -169,8 +490,8 @@ static void pnp_assign_dma(struct pnp_de
1, 3, 5, 6, 7, 0, 2, 4
};

- if (idx >= PNP_MAX_DMA) {
- dev_err(&dev->dev, "too many DMA resources\n");
+ if (!pnp_dma_ok(dev, idx)) {
+ pnp_print_assign_err("DMA", idx);
return;
}

@@ -207,28 +528,28 @@ void pnp_init_resource_table(struct pnp_
{
int idx;

- for (idx = 0; idx < PNP_MAX_IRQ; idx++) {
+ for (idx = 0; idx < table->allocated_irqs; idx++) {
table->irq_resource[idx].name = NULL;
table->irq_resource[idx].start = -1;
table->irq_resource[idx].end = -1;
table->irq_resource[idx].flags =
IORESOURCE_IRQ | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
- for (idx = 0; idx < PNP_MAX_DMA; idx++) {
+ for (idx = 0; idx < table->allocated_dmas; idx++) {
table->dma_resource[idx].name = NULL;
table->dma_resource[idx].start = -1;
table->dma_resource[idx].end = -1;
table->dma_resource[idx].flags =
IORESOURCE_DMA | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
- for (idx = 0; idx < PNP_MAX_PORT; idx++) {
+ for (idx = 0; idx < table->allocated_ports; idx++) {
table->port_resource[idx].name = NULL;
table->port_resource[idx].start = 0;
table->port_resource[idx].end = 0;
table->port_resource[idx].flags =
IORESOURCE_IO | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
}
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
– for (idx = 0; idx < PNP_MAX_MEM; idx++) {
+ for (idx = 0; idx < table->allocated_mems; idx++) {
table->mem_resource[idx].name = NULL;
table->mem_resource[idx].start = 0;
table->mem_resource[idx].end = 0;
@@ -245,7 +566,7 @@ static void pnp_clean_resource_table(str
{
int idx;

– for (idx = 0; idx < PNP_MAX_IRQ; idx++) {
+ for (idx = 0; idx < res->allocated_irqs; idx++) {
if (!(res->irq_resource[idx].flags & IORESOURCE_AUTO))
continue;
res->irq_resource[idx].start = -1;
@@ -253,7 +574,7 @@ static void pnp_clean_resource_table(str
res->irq_resource[idx].flags =
IORESOURCE_IRQ | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
– for (idx = 0; idx < PNP_MAX_DMA; idx++) {
+ for (idx = 0; idx < res->allocated_dmas; idx++) {
if (!(res->dma_resource[idx].flags & IORESOURCE_AUTO))
continue;
res->dma_resource[idx].start = -1;
@@ -261,7 +582,7 @@ static void pnp_clean_resource_table(str
res->dma_resource[idx].flags =
IORESOURCE_DMA | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
– for (idx = 0; idx < PNP_MAX_PORT; idx++) {
+ for (idx = 0; idx < res->allocated_ports; idx++) {
if (!(res->port_resource[idx].flags & IORESOURCE_AUTO))
continue;
res->port_resource[idx].start = 0;
@@ -269,7 +590,7 @@ static void pnp_clean_resource_table(str
res->port_resource[idx].flags =
IORESOURCE_IO | IORESOURCE_AUTO | IORESOURCE_UNSET;
}
– for (idx = 0; idx < PNP_MAX_MEM; idx++) {
+ for (idx = 0; idx < res->allocated_mems; idx++) {
if (!(res->mem_resource[idx].flags & IORESOURCE_AUTO))
continue;
res->mem_resource[idx].start = 0;
@@ -386,46 +707,12 @@ fail:
int pnp_manual_config_dev(struct pnp_dev *dev, struct pnp_resource_table *res,
int mode)
{
– int i;
– struct pnp_resource_table *bak;
–
– if (!pnp_can_configure(dev))
– return -ENODEV;
– bak = pnp_alloc(sizeof(struct pnp_resource_table));
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
- if (!bak)
- return -ENOMEM;
- *bak = dev->res;
-
- down(&pnp_res_mutex);
- dev->res = *res;
- if (!(mode & PNP_CONFIG_FORCE)) {
- for (i = 0; i < PNP_MAX_PORT; i++) {
- if (!pnp_check_port(dev, i))
- goto fail;
- }
- for (i = 0; i < PNP_MAX_MEM; i++) {
- if (!pnp_check_mem(dev, i))
- goto fail;
- }
- for (i = 0; i < PNP_MAX_IRQ; i++) {
- if (!pnp_check_irq(dev, i))
- goto fail;
- }
- for (i = 0; i < PNP_MAX_DMA; i++) {
- if (!pnp_check_dma(dev, i))
- goto fail;
- }
- }
- up(&pnp_res_mutex);
-
- kfree(bak);
- return 0;
+ /* We must never end up here, these functions are poison for dynamic
+ allocation via pointer array.
+ */
+ BUG_ON(1);
+ return 1;

-fail:
- dev->res = *bak;
- up(&pnp_res_mutex);
- kfree(bak);
- return -EINVAL;
}

/**
@@ -563,6 +850,7 @@ int pnp_disable_dev(struct pnp_dev *dev)
void pnp_resource_change(struct resource *resource, resource_size_t start,
resource_size_t size)
{
+ return;
resource->flags &= ~(IORESOURCE_AUTO | IORESOURCE_UNSET);
resource->start = start;
resource->end = start + size - 1;
Index: linux-2.6.23/drivers/pnp/pnpacpi/rsparser.c
```

```

=====
--- linux-2.6.23.orig/drivers/pnp/pnpacpi/rsparser.c
+++ linux-2.6.23/drivers/pnp/pnpacpi/rsparser.c
@@ -73,23 +73,16 @@ static void pnpacpi_parse_allocated_irqr
u32 gsi, int triggering,
int polarity, int shareable)
{
- int i = 0;
int irq;
int p, t;
- static unsigned char warned;
+
+ struct resource new_res = {
+ .flags = IORESOURCE_IRQ,
+ };

if (!valid_IRQ(gsi))
return;

- while (!(res->irq_resource[i].flags & IORESOURCE_UNSET) &&
- i < PNP_MAX_IRQ)
- i++;
- if (i >= PNP_MAX_IRQ && !warned) {
- printk(KERN_ERR "pnpacpi: exceeded the max number of IRQ "
- "resources: %d \n", PNP_MAX_IRQ);
- warned = 1;
- return;
- }
/*
* in IO-APIC mode, use overridden attribute. Two reasons:
* 1. BIOS bug in DSDT
@@ -107,20 +100,26 @@ static void pnpacpi_parse_allocated_irqr
}
}

- res->irq_resource[i].flags = IORESOURCE_IRQ; // Also clears _UNSET flag
- res->irq_resource[i].flags |= irq_flags(triggering, polarity);
+ new_res.flags |= irq_flags(triggering, polarity);
irq = acpi_register_gsi(gsi, triggering, polarity);
if (irq < 0) {
- res->irq_resource[i].flags |= IORESOURCE_DISABLED;
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
return;
}

if (shareable)
- res->irq_resource[i].flags |= IORESOURCE_IRQ_SHAREABLE;

```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
+ new_res.flags |= IORESOURCE_IRQ_SHAREABLE;

- res->irq_resource[i].start = irq;
- res->irq_resource[i].end = irq;
+ new_res.start = irq;
+ new_res.end = irq;
pcibios_penalize_isa_irq(irq, 1);
+
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}

static int dma_flags(int type, int bus_master, int transfer)
@@ -170,81 +169,71 @@ static void pnpacpi_parse_allocated_dmar
u32 dma, int type,
int bus_master, int transfer)
{
- int i = 0;
- static unsigned char warned;
-
- while (i < PNP_MAX_DMA &&
- !(res->dma_resource[i].flags & IORESOURCE_UNSET))
- i++;
- if (i < PNP_MAX_DMA) {
- res->dma_resource[i].flags = IORESOURCE_DMA; // Also clears _UNSET flag
- res->dma_resource[i].flags |=
- dma_flags(type, bus_master, transfer);
- if (dma == -1) {
- res->dma_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->dma_resource[i].start = dma;
- res->dma_resource[i].end = dma;
- } else if (!warned) {
- printk(KERN_ERR "pnpacpi: exceeded the max number of DMA "
- "resources: %d\n", PNP_MAX_DMA);
- warned = 1;
- }
+ struct resource new_res = {
+ .flags = IORESOURCE_DMA,
+ };
+
+ new_res.flags |= dma_flags(type, bus_master, transfer);
+ if (dma == -1) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
}
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
+ new_res.start = dma;
+ new_res.end = dma;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
```

```
static void pnpacpi_parse_allocated_ioresource(struct pnp_resource_table *res,
u64 io, u64 len, int io_decode)
{
- int i = 0;
- static unsigned char warned;
-
- while (!(res->port_resource[i].flags & IORESOURCE_UNSET) &&
- i < PNP_MAX_PORT)
- i++;
- if (i < PNP_MAX_PORT) {
- res->port_resource[i].flags = IORESOURCE_IO; // Also clears _UNSET flag
- if (io_decode == ACPI_DECODE_16)
- res->port_resource[i].flags |= PNP_PORT_FLAG_16BITADDR;
- if (len <= 0 || (io + len - 1) >= 0x10003) {
- res->port_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->port_resource[i].start = io;
- res->port_resource[i].end = io + len - 1;
- } else if (!warned) {
- printk(KERN_ERR "pnpacpi: exceeded the max number of IO "
- "resources: %d \n", PNP_MAX_PORT);
- warned = 1;
- }
+ struct resource new_res = {
+ .flags = IORESOURCE_IO,
+ };
+
+ if (io_decode == ACPI_DECODE_16)
+ new_res.flags |= PNP_PORT_FLAG_16BITADDR;
+ if (len <= 0 || (io + len - 1) >= 0x10003) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = io;
+ new_res.end = io + len - 1;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
```

```
static void pnpacpi_parse_allocated_memresource(struct pnp_resource_table *res,
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
u64 mem, u64 len,
int write_protect)
{
- int i = 0;
- static unsigned char warned;
-
- while (!(res->mem_resource[i].flags & IORESOURCE_UNSET) &&
- (i < PNP_MAX_MEM))
- i++;
- if (i < PNP_MAX_MEM) {
- res->mem_resource[i].flags = IORESOURCE_MEM; // Also clears _UNSET flag
- if (len <= 0) {
- res->mem_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- if (write_protect == ACPI_READ_WRITE_MEMORY)
- res->mem_resource[i].flags |= IORESOURCE_MEM_WRITEABLE;
-
- res->mem_resource[i].start = mem;
- res->mem_resource[i].end = mem + len - 1;
- } else if (!warned) {
- printk(KERN_ERR "pnpacpi: exceeded the max number of mem "
- "resources: %d\n", PNP_MAX_MEM);
- warned = 1;
- }
+ struct resource new_res = {
+ .flags = IORESOURCE_MEM,
+ };
+
+ if (len <= 0) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ if (write_protect == ACPI_READ_WRITE_MEMORY)
+ new_res.flags |= IORESOURCE_MEM_WRITEABLE;
+
+ new_res.start = mem;
+ new_res.end = mem + len - 1;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }

static void pnpacpi_parse_allocated_address_space(struct pnp_resource_table *res_table,
Index: linux-2.6.23/drivers/pnp/pnpbios/rsparser.c
=====
--- linux-2.6.23.orig/drivers/pnp/pnpbios/rsparser.c
+++ linux-2.6.23/drivers/pnp/pnpbios/rsparser.c
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
@@ -56,78 +56,84 @@ inline void pcibios_penalize_isa_irq(int
static void pnpbios_parse_allocated_irqresource(struct pnp_resource_table *res,
int irq)
{
- int i = 0;
-
- while (!(res->irq_resource[i].flags & IORESOURCE_UNSET)
- && i < PNP_MAX_IRQ)
- i++;
- if (i < PNP_MAX_IRQ) {
- res->irq_resource[i].flags = IORESOURCE_IRQ; // Also clears _UNSET flag
- if (irq == -1) {
- res->irq_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->irq_resource[i].start =
- res->irq_resource[i].end = (unsigned long)irq;
- pcibios_penalize_isa_irq(irq, 1);
+ struct resource new_res = {
+ .flags = IORESOURCE_IRQ,
+ };
+
+ if (irq == -1) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = new_res.end = (unsigned long)irq;
+ pcibios_penalize_isa_irq(irq, 1);
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }

static void pnpbios_parse_allocated_dmaresource(struct pnp_resource_table *res,
int dma)
{
- int i = 0;
-
- while (i < PNP_MAX_DMA &&
- !(res->dma_resource[i].flags & IORESOURCE_UNSET))
- i++;
- if (i < PNP_MAX_DMA) {
- res->dma_resource[i].flags = IORESOURCE_DMA; // Also clears _UNSET flag
- if (dma == -1) {
- res->dma_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->dma_resource[i].start =
```

## Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
- res->dma_resource[i].end = (unsigned long)dma;
+ struct resource new_res = {
+ .flags = IORESOURCE_DMA,
+ };
+
+ if (dma == -1) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = new_res.end = (unsigned long)dma;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }

static void pnpbios_parse_allocated_ioresource(struct pnp_resource_table *res,
int io, int len)
{
- int i = 0;
-
- while (!(res->port_resource[i].flags & IORESOURCE_UNSET)
- && i < PNP_MAX_PORT)
- i++;
- if (i < PNP_MAX_PORT) {
- res->port_resource[i].flags = IORESOURCE_IO; // Also clears _UNSET flag
- if (len <= 0 || (io + len - 1) >= 0x10003) {
- res->port_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->port_resource[i].start = (unsigned long)io;
- res->port_resource[i].end = (unsigned long)(io + len - 1);
+ struct resource new_res = {
+ .flags = IORESOURCE_IO,
+ };
+
+ if (len <= 0 || (io + len - 1) >= 0x10003) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = (unsigned long)io;
+ new_res.end = (unsigned long)(io + len - 1);
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }
}
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
static void pnpbios_parse_allocated_memresource(struct pnp_resource_table *res,
int mem, int len)
{
- int i = 0;
-
- while (!(res->mem_resource[i].flags & IORESOURCE_UNSET)
- && i < PNP_MAX_MEM)
- i++;
- if (i < PNP_MAX_MEM) {
- res->mem_resource[i].flags = IORESOURCE_MEM; // Also clears _UNSET flag
- if (len <= 0) {
- res->mem_resource[i].flags |= IORESOURCE_DISABLED;
- return;
- }
- res->mem_resource[i].start = (unsigned long)mem;
- res->mem_resource[i].end = (unsigned long)(mem + len - 1);
+ struct resource new_res = {
+ .flags = IORESOURCE_MEM,
+ };
+
+ if (len <= 0) {
+ /* <trenn> Check: Do we need to allocate and assign
+ this resource at all? */
+ new_res.flags |= IORESOURCE_DISABLED;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ return;
+ }
+ new_res.start = (unsigned long)mem;
+ new_res.end = (unsigned long)(mem + len - 1);
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
+ }
}
```

```
static unsigned char *pnpbios_parse_allocated_resource_data(unsigned char *p,
Index: linux-2.6.23/drivers/pnp/core.c
```

```
=====
--- linux-2.6.23.orig/drivers/pnp/core.c
+++ linux-2.6.23/drivers/pnp/core.c
@@ -129,6 +129,7 @@ int __pnp_add_device(struct pnp_dev *dev
return ret;
```

```
pnp_interface_attach_device(dev);
+ pnp_dump_resources(dev);
return 0;
}
```

```
@@ -148,6 +149,7 @@ int pnp_add_device(struct pnp_dev *dev)
dev->dev.parent = &dev->protocol->dev;
sprintf(dev->dev.bus_id, "%02x:%02x", dev->protocol->number,
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
dev->number);
+ pnp_dbg("Adding device %s - %s", dev->name, dev->dev.bus_id);
ret = __pnp_add_device(dev);
if (ret)
return ret;
Index: linux-2.6.23/drivers/pnp/isapnp/core.c
=====
--- linux-2.6.23.orig/drivers/pnp/isapnp/core.c
+++ linux-2.6.23/drivers/pnp/isapnp/core.c
@@ -42,6 +42,7 @@
#include <linux/init.h>
#include <linux/isapnp.h>
#include <linux/mutex.h>
+
#include <asm/io.h>

#if 0
@@ -65,6 +66,12 @@ module_param(isapnp_verbose, int, 0);
MODULE_PARM_DESC(isapnp_verbose, "ISA Plug & Play verbose mode");
MODULE_LICENSE("GPL");

+/* ISAPNP is restricted to these limits by spec */
+#define PNP_MAX_PORT 8
+#define PNP_MAX_MEM 4
+#define PNP_MAX_IRQ 2
+#define PNP_MAX_DMA 2
+
#define _PIDXR 0x279
#define _PNPWRP 0xa79

@@ -942,6 +949,7 @@ static int isapnp_read_resources(struct
struct pnp_resource_table *res)
{
int tmp, ret;
+ struct resource new_res;

dev->active = isapnp_read_byte(ISAPNP_CFG_ACTIVATE);
if (dev->active) {
@@ -949,16 +957,20 @@ static int isapnp_read_resources(struct
ret = isapnp_read_word(ISAPNP_CFG_PORT + (tmp << 1));
if (!ret)
continue;
- res->port_resource[tmp].start = ret;
- res->port_resource[tmp].flags = IORESOURCE_IO;
+ new_res.start = ret;
+ new_res.flags = IORESOURCE_IO;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
for (tmp = 0; tmp < PNP_MAX_MEM; tmp++) {
ret =
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
isapnp_read_word(ISAPNP_CFG_MEM + (tmp << 3)) << 8;
if (!ret)
continue;
- res->mem_resource[tmp].start = ret;
- res->mem_resource[tmp].flags = IORESOURCE_MEM;
+ new_res.start = ret;
+ new_res.flags = IORESOURCE_MEM;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
for (tmp = 0; tmp < PNP_MAX_IRQ; tmp++) {
ret =
@@ -966,17 +978,20 @@ static int isapnp_read_resources(struct
8);
if (!ret)
continue;
- res->irq_resource[tmp].start =
- res->irq_resource[tmp].end = ret;
- res->irq_resource[tmp].flags = IORESOURCE_IRQ;
+ new_res.start = new_res.end = ret;
+ new_res.flags = IORESOURCE_IRQ;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
for (tmp = 0; tmp < PNP_MAX_DMA; tmp++) {
ret = isapnp_read_byte(ISAPNP_CFG_DMA + tmp);
if (ret == 4)
continue;
res->dma_resource[tmp].start =
- res->dma_resource[tmp].end = ret;
- res->dma_resource[tmp].flags = IORESOURCE_DMA;
+ new_res.start = new_res.end = ret;
+ new_res.flags = IORESOURCE_DMA;
+ if (pnp_assign_resource(res, &new_res))
+ pnp_err("Bug in %s", __FUNCTION__);
}
}
return 0;
```

Index: linux-2.6.23/drivers/pnp/resource.c

```
=====
--- linux-2.6.23.orig/drivers/pnp/resource.c
+++ linux-2.6.23/drivers/pnp/resource.c
@@ -242,7 +242,7 @@ int pnp_check_port(struct pnp_dev *dev,
}

/* check for internal conflicts */
- for (tmp = 0; tmp < PNP_MAX_PORT && tmp != idx; tmp++) {
+ for (tmp = 0; pnp_port_ok(dev, tmp) && tmp != idx; tmp++) {
if (dev->res.port_resource[tmp].flags & IORESOURCE_IO) {
tport = &dev->res.port_resource[tmp].start;
tend = &dev->res.port_resource[tmp].end;
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
@@ -255,7 +255,7 @@ int pnp_check_port(struct pnp_dev *dev,
pnp_for_each_dev(tdev) {
if (tdev == dev)
continue;
- for (tmp = 0; tmp < PNP_MAX_PORT; tmp++) {
+ for (tmp = 0; pnp_port_ok(tdev, tmp); tmp++) {
if (tdev->res.port_resource[tmp].flags & IORESOURCE_IO) {
if (cannot_compare
(tdev->res.port_resource[tmp].flags))
@@ -300,7 +300,7 @@ int pnp_check_mem(struct pnp_dev *dev, i
}

/* check for internal conflicts */
- for (tmp = 0; tmp < PNP_MAX_MEM && tmp != idx; tmp++) {
+ for (tmp = 0; pnp_mem_ok(dev, tmp) && tmp != idx; tmp++) {
if (dev->res.mem_resource[tmp].flags & IORESOURCE_MEM) {
taddr = &dev->res.mem_resource[tmp].start;
tend = &dev->res.mem_resource[tmp].end;
@@ -313,7 +313,7 @@ int pnp_check_mem(struct pnp_dev *dev, i
pnp_for_each_dev(tdev) {
if (tdev == dev)
continue;
- for (tmp = 0; tmp < PNP_MAX_MEM; tmp++) {
+ for (tmp = 0; pnp_mem_ok(tdev, tmp); tmp++) {
if (tdev->res.mem_resource[tmp].flags & IORESOURCE_MEM) {
if (cannot_compare
(tdev->res.mem_resource[tmp].flags))
@@ -355,7 +355,7 @@ int pnp_check_irq(struct pnp_dev *dev, i
}

/* check for internal conflicts */
- for (tmp = 0; tmp < PNP_MAX_IRQ && tmp != idx; tmp++) {
+ for (tmp = 0; pnp_irq_ok(dev, tmp) && tmp != idx; tmp++) {
if (dev->res.irq_resource[tmp].flags & IORESOURCE_IRQ) {
if (dev->res.irq_resource[tmp].start == *irq)
return 0;
@@ -388,7 +388,7 @@ int pnp_check_irq(struct pnp_dev *dev, i
pnp_for_each_dev(tdev) {
if (tdev == dev)
continue;
- for (tmp = 0; tmp < PNP_MAX_IRQ; tmp++) {
+ for (tmp = 0; pnp_irq_ok(tdev, tmp); tmp++) {
if (tdev->res.irq_resource[tmp].flags & IORESOURCE_IRQ) {
if (cannot_compare
(tdev->res.irq_resource[tmp].flags))
@@ -424,7 +424,7 @@ int pnp_check_dma(struct pnp_dev *dev, i
}

/* check for internal conflicts */
- for (tmp = 0; tmp < PNP_MAX_DMA && tmp != idx; tmp++) {
+ for (tmp = 0; pnp_dma_ok(dev, tmp) && tmp != idx; tmp++) {
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

```
if (dev->res.dma_resource[tmp].flags & IORESOURCE_DMA) {
if (dev->res.dma_resource[tmp].start == *dma)
return 0;
@@ -443,7 +443,7 @@ int pnp_check_dma(struct pnp_dev *dev, i
pnp_for_each_dev(tdev) {
if (tdev == dev)
continue;
- for (tmp = 0; tmp < PNP_MAX_DMA; tmp++) {
+ for (tmp = 0; pnp_dma_ok(tdev, tmp); tmp++) {
if (tdev->res.dma_resource[tmp].flags & IORESOURCE_DMA) {
if (cannot_compare
(tdev->res.dma_resource[tmp].flags))
Index: linux-2.6.23/drivers/pnp/support.c
```

```
----- linux-2.6.23.orig/drivers/pnp/support.c
+++ linux-2.6.23/drivers/pnp/support.c
@@ -14,11 +14,16 @@
* resources
* @dev: pointer to the desired PnP device
*/
+
+/* <trenn> This interface is only used by pnpbios and one driver:
+ sound/isa/sscape.c
+ drivers can check for pnp_port_valid... anyway
+ This one is not needed.
+*/
int pnp_is_active(struct pnp_dev *dev)
{
- if (!pnp_port_start(dev, 0) && pnp_port_len(dev, 0) <= 1 &&
- !pnp_mem_start(dev, 0) && pnp_mem_len(dev, 0) <= 1 &&
- pnp_irq(dev, 0) == -1 && pnp_dma(dev, 0) == -1)
+ if (dev->res.allocated_ports <= 0 && dev->res.allocated_mems <= 0 &&
+ dev->res.allocated_irqs <= 0 && dev->res.allocated_dmas <= 0)
return 0;
else
return 1;
Index: linux-2.6.23/drivers/pnp/system.c
```

```
----- linux-2.6.23.orig/drivers/pnp/system.c
+++ linux-2.6.23/drivers/pnp/system.c
@@ -58,7 +58,7 @@ static void reserve_resources_of_dev(str
{
int i;

- for (i = 0; i < PNP_MAX_PORT; i++) {
+ for (i = 0; pnp_port_ok(dev, i); i++) {
if (!pnp_port_valid(dev, i))
continue;
if (pnp_port_start(dev, i) == 0)
@@ -80,7 +80,7 @@ static void reserve_resources_of_dev(str
pnp_port_end(dev, i), 1);
```

Re: [PATCH] Allocate pnp resources dynamically via krealloc – working version

}

```
- for (i = 0; i < PNP_MAX_MEM; i++) {  
+ for (i = 0; pnp_mem_ok(dev, i); i++) {  
if (!pnp_mem_valid(dev, i))  
continue;
```

--

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

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

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